NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

A PROTOTYPE EXPERT SYSTEM WHICH ASSIGNS AVIATION MAINTENANCE PERSONNEL TO SQUADRON BILLETS

by

Thomas Porter Alston

March 1987

Thesis Advisor: Norman R. Lyons

unclassified
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a REPORT SECURITY CLASSIFICATION unclassified	1b. RESTRICTIVE MARKINGS
2a SECURITY CLASSIFICATION AUTHORITY	3 DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution is unlimited.
2b DECLASSIFICATION/DOWNGRADING SCHEDULE	
4 PERFORMING ORGANIZATION REPORT NUMBER(S)	5 MONITORING ORGANIZATION REPORT NUMBER(S)
6a NAME OF PERFORMING ORGANIZATION Naval Postgraduate School; 6b OFFICE SYMBOL (If applicable) 54	7a NAME OF MONITORING ORGANIZATION Naval Postgraduate School
6c ADDRESS (City, State, and ZIP Code) Monterey, California 93943-5000	7b ADDRESS (City, State, and ZIP Code) Monterey, California 93943-5000
8a NAME OF FUNDING/SPONSORING ORGANIZATION; 8b OFFICE SYMBOL (If applicable)	9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c ADDRESS (City, State, and ZIP Code)	10 SOURCE OF FUNDING NUMBERS: PROGRAM ELEMENT NO; PROJECT NO; TASK NO; WORK UNIT ACCESSION NO

11 TITLE (Include Security Classification) A PROTOTYPE EXPERT SYSTEM WHICH ASSIGNS AVIATION MAINTENANCE PERSONNEL TO SQUADRON BILLETS

12 PERSONAL AUTHOR(S) Alston, Thomas Porter

13a TYPE OF REPORT	13b TIME COVERED	14 DATE OF REPORT (Year, Month, Day)	15 PAGE COUNT
Master's Thesis	FROM ______ TO ______	1987 March	119

16 SUPPLEMENTARY NOTATION

17 COSATI CODES			18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	expert systems, decision support system, aviation maintenance, heuristics.

19 ABSTRACT (Continue on reverse if necessary and identify by block number)

The technical feasibility of developing an expert system to support the assignment of squadron maintenance personnel to authorized billets is examined by building a prototype system. The rules used by an assistant maintenance officer to assign personnel are analyzed, and a database derived from the OPNAV 1000/2 and EDVR is designed. The prototype is developed for a micro-computer system using an expert system shell, Insight 2+, and is fully integrated with dBase III files.

20 DISTRIBUTION/AVAILABILITY OF ABSTRACT [✓ UNCLASSIFIED/UNLIMITED] ☐ SAME AS RPT ☐ DTIC USERS	21 ABSTRACT SECURITY CLASSIFICATION unclassified	
22a NAME OF RESPONSIBLE INDIVIDUAL Prof. Norman R. Lyons	22b TELEPHONE (Include Area Code) (408) 646-2666	22c OFFICE SYMBOL Code 54Lb

DD FORM 1473, 84 MAR — 83 APR edition may be used until exhausted. All other editions are obsolete

SECURITY CLASSIFICATION OF THIS PAGE
unclassified

A Prototype Expert System which Assigns Aviation Maintenance Personnel to Squadron Billets

by

Thomas Porter Alston
Lieutenant, United States Navy
B.S., University of Utah, 1974

Submitted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE IN INFORMATION SYSTEMS

from the

NAVAL POSTGRADUATE SCHOOL
March 1987

Author: Thomas Porter Alston
Thomas Porter Alston

Approved by: [illegible]
N. R. Lyons, Thesis Advisor

[illegible]
T. Sivasankaran, Second Reader

[illegible]
W. R. Greer, Jr., Chairman,
Department of Administrative Sciences

Kneale T. Marshall
Kneale T. Marshall,
Dean of Information and Policy Sciences

ABSTRACT

The technical feasibility of developing an expert system to support the assignment of squadron maintenance personnel to authorized billets is examined by building a prototype system. The rules used by an assistant maintenance officer to assign personnel are analyzed, and a database derived from the OPNAV 1000/2 and EDVR is designed. The prototype is developed for a micro-computer system using an expert system shell, Insight 2+, and is fully integrated with dBase III files.

THESIS DISCLAIMER

TABLE OF CONTENTS

I. INTRODUCTION

One of the duties of the assistant maintenance officer in an aviation squadron is to assign maintenance personnel to the various squadron maintenance billets. Specifically, the NAMP [Ref. 1] defines his responsibility as:

> Determine the apportionment of maintenance personnel assigned to the department and monitor/coordinate the assignment of Temporary Additional Duty (TAD) personnel to other activities.

The assistant maintenance officer must take into account details such as when personnel are leaving the squadron, personal qualifications, training and duration of the job.

The purpose of this thesis is to demonstrate that it is technically feasible to build an expert system which assigns aviation maintenance personnel to appropriate billets. The technical feasibility will be demonstrated by building a rule-based expert system prototype which accesses database files to determine how to assign personnel to billets. The required inputs consist of two database files, one which contains all the squadron billets, and the other holds all the squadron personnel information. The output is a file containing the billet, the person to be assigned to the billet, and, when required, the prospective person who will fill the billet in thirty days or less.

The prototype differs from expert systems which search several different alternatives until an acceptable one is found. Instead, it provides a single acceptable solution. It also differs from most expert systems in that the user interaction is done almost exclusively through database files with minimal interaction with the user. To verify the expert system under realistic conditions, simulated databases of billets and personnel from an F/A-18 squadron were created.

Chapter II provides a statement of the problem definition. It also defines the scope, objectives and constraints in building the expert system. Chapter III describes the analysis of the problem and the initial rules considered for the prototype. Chapter IV addresses the implementation of the expert system. Chapter V discusses recommendations for further modifications of the prototype system. Chapter VI presents the conclusion of this project.

II. PROBLEM DEFINITION

The assistant maintenance officer rotates out of the squadron at least every three years, taking with him the knowledge and expertise he has accumulated. He is usually replaced by a less experienced person, and it can take up to six months to get the person trained to the point where he manages the functions of assigning personnel and scheduling training efficiently and effectively.

The assistant maintenance officer uses information from several sources to make decisions as to where to assign personnel. One of the main documents is the Manpower Authorization (OPNAV 1000/2) which is the military manpower requirements authorized by the Chief of Naval Operations (CNO). It is the official organization manning and billet authorization for a squadron. It identifies the billet sequence code, the billet/position title, designated rate, Navy Enlisted Classification (NEC) code, the number of persons authorized (basic allowance), and the planned authorizations for the next five fiscal years.

Another primary document is the Enlisted Distribution and Verification Report (EDVR). This report reflects all individuals assigned to the activity, including prospective gains. It contains a roster of all personnel, and includes their social security numbers, rates, NEC's, date assigned to the squadron, projected rotation date (PRD), and the end of active obligated service date (EAOS).

The assistant maintenance officer must also know what temporary additional duty (TAD) billets have to be filled. The TAD requirements vary depending on whether the squadron is stationed at a Naval Air Station (NAS) or on board a carrier. The specific requirements are spelled out in the appropriate NAS, wing or ship instructions. Most of the TAD billets are of ninety days duration and usually only junior personnel (E1 through E3) are assigned to them.

Other factors which must be taken into account when assigning personnel are special qualifications a person might have. These include collateral duty inspector (CDI), quality assurance representative (QAR), plane captain (PC), and ordnance certification. The assistant maintenance officer by virtue of his position in the squadron should simply know what qualifications each person has. The information can be also obtained from the monthly maintenance plan.

A. OBJECTIVES

The objective is to build a prototype expert system which can assign personnel to billets. Its utility is that it will improve the assistant maintenance officer's decision making by offering an "expert" solution to the problem. It should also decrease the time it takes to train a new person to administer the program. Solving this problem is important because the assignment of personnel to billets and tracking of TAD personnel is one of the assistant maintenance officer's more time consuming activities. It involves numerous variables and constantly changing data, and access to an "expert" consultant would allow him to spend more time on squadron training requirements and other personnel matters.

B. SCOPE

This expert system application is limited to the squadron maintenance environment and it emphasizes automation of one of the assistant maintenance officer's functions. It is limited to the assignment of persons to suitable billets within certain constraints, and that the assignments it recommends are as good as the assignments made by an experienced person. The assignment of personnel is very subjective, and even acknowledged experts can have wide differences of opinions. Therefore, it is not possible to prove that any particular assignment of personnel is the optimum assignment, and there is no attempt to claim that the expert system generated solution is an optimal solution.

The project is further limited to application on a personal computer since most squadrons have micro-computers and access to larger machines on board ships is unlikely. In addition, relatively inexpensive software for expert system shells and database managers is readily available for micro-computers.

C. CONSTRAINTS

1. Hardware

The computer hardware selected for this project is an IBM-PC or compatible machine. It was selected because of the availability of several commercial products for expert systems and database managers in the MS-DOS operating environment. The minimum requirements are 512K internal memory and two disk drives. A hard disk drive is not necessary, however it is highly desirable. The prototype was built and tested on an IBM-AT.

2. Software

Waterman [Ref. 2] suggests several questions to consider when choosing the expert system tool. Does it have the power and sophistication needed? Are the support facilities adequate, especially when considering time constraints? Is it reliable and is it maintained? Does it have the features that will be required to solve the problem?

Insight 2+ was selected as the expert system tool primarily because of two features: its capability to access dBase III files and its user-friendliness. Insight 2+ has a built-in programming language (DBPAS) which allows the knowledge base to access dBase III files. Insight's similarities to Turbo Pascal and Wordstar make the system easy to learn. Insight 2+ comes with an ASCII parameter passing program written in Turbo Pascal which allows the user to write Turbo Pascal programs and pass parameters between the knowledge base and a compiled Turbo Pascal program. Also, the DBPAS programming language is very similar to Turbo Pascal with some modifications for the dBase III functions. The text editor in Insight 2+ uses the Wordstar command set for editing both knowledge base programs and DBPAS programs.

Insight's capabilities are limited to development of small expert systems of less than 400 rules. The development of this prototype was expected to be in the range of 100 rules, hence Insight was more than adequate to meet the requirements.

dBase III was selected for this project because of its compatibility with Insight 2+. dBase III is a relational database, and has the capability of handling a typical aviation squadron database of approximately 200 personnel. An additional consideration in choosing dBase III was my familiarity with it.

III. ANALYSIS

The goal of the analysis phase was to determine how the problem should be solved. The first step in the process was to study the present system and obtain a good understanding of the physical system. The second step was to develop the initial rules that would form the basis for building the knowledge base. The last step was to define the data files and data elements used by the knowledge base.

A. CURRENT PHYSICAL SYSTEM

In the current system, the squadron assistant maintenance officer uses information from several different sources before making personnel decisions. The information that is routinely used to determine which billets are authorized comes from the Manpower Authorization (OPNAV 1000/2) and local instructions which specify TAD requirements. The information from the OPNAV 1000/2 required by the knowledge base is the billet/position title, designated rate, Navy Enlisted Classification (NEC) code, and the number of persons authorized. The TAD billet information required by the knowledge base includes the billet description, number of days the TAD tour lasts, and the required paygrade.

The information that determincs personnel on board and their qualifications comes from the EDVR and from internal squadron sources. The EDVR information used to assign personnel is the list of personnel and prospective gains, rate, NEC, date assigned to the squadron, PRD, and EAOS. Special qualifications include CDI, QAR, PC, and ordnance certification.

One approach that an assistant maintenance officer takes is to review each billet looking for personnel changes. If a person is currently filling a billet (which is not a TAD assignment) and his PRD/EAOS are not within the next thirty days, then that person will remain in the same billet. However, if a person will be leaving the squadron in the next thirty days because of PRD or EAOS, then a replacement has to be found. In the case of TAD billets, if a person's time in the billet exceeds ninety days, then a replacement has to be found.

The assistant maintenance officer determines the requirements necessary to fill a position by looking at the work center and knowing what rates can be assigned to that work center. If the billet is in work center 110, the power plants branch, then an "AD" or aviation machinists mate is required. If the billet is in work center 120, airframes branch, then both "AMH", aviation structural hydraulics, and "AMS", aviation structural mechanic, personnel are required. In work center 12C, corrosion control branch, "AMS" personnel are assigned. In the aviation life support systems branch, work center 13A requires parachute riggers, "PR" rating, and work center 13B requires "AME", aviation structural mechanic egress, personnel.

Aviation electronics technicians, "AT" rating, are assigned to work center 210, the electronics branch, while aviation electrician mates, "AE", are assigned to work center 220, the electrical/instrument branch. The armament branch, work center 230, has "AO", aviation ordnanceman, personnel. If a billet is in work center 260, the radar/fire control branch, then an "AQ", aviation fire control technician is assigned. In the line division, work centers 310, plane captains branch, and 320 troubleshooter branch, persons of any rating can be assigned.

Administrative personnel include "AZ", aviation administrationman, and "AK", aviation storekeepers, personnel. AZ's are assigned to work center 04C as the data analyst, work center 04A as the Quality Assurance librarian, work center 020 as a maintenance control clerk, or work center 030 as a maintenance administration clerk. AK's are assigned to the material control branch, work center 050.

Other considerations in assigning personnel are their special qualifications. A quality assurance representative (QAR) should be a senior petty officer, preferably E6 or above, and have a well-rounded maintenance background. They are responsible for performing inspections of critical maintenance tasks which require a final quality assurance inspection. They also monitor the performance of collateral duty inspectors. QAR's are assigned to the quality assurance division, work center 040, and typically one person from each production work center is required. For example, a squadron will usually have one QAR from each of the following ratings: AD, AMH or AMS, AME, AT, AE, AO and AQ. It is also possible to temporarily assign a person from a work center on a collateral duty basis, CDQAR, however, for this prototype CDQAR's were not considered. A collateral duty inspector (CDI) is another special qualification. CDI's are required to inspect all work and comply with quality assurance inspections required for the maintenance performed in their work center. CDI's are screened very carefully before being designated as a CDI. Because of the time it takes to qualify personnel as a CDI or QAR, one of the assistant maintenance officer's priorities is to keep them in the corresponding work center.

Other qualifications ordnance certification for personnel in work center "230" and plane captain designation in work center "310". Usually, personnel who have these qualifications should be assigned to that work center.

This is the beginning of the process to translate the on the job knowledge to rules for the expert system. In the next section, the initial rules will be developed based on reasoning similar to the above.

B. DEVELOPING THE INITIAL RULES

In developing the initial rules, there are two conditions which have to be met before the knowledge base has completed its task. The first condition is that the knowledge base has checked each billet in the billet database and a person has been assigned to the billet. The second condition is that the knowledge base has verified each person in the personnel database is assigned to a billet. To put the rule in a format similar to Insight 2+, it will read:

```
IF all billets are filled
AND all personnel are assigned
THEN AMO is done
```

To fill a single billet the requirements are to go to the billet database and get the next billet, then search the personnel database and find a person who meets the billet requirements. Also, the knowledge base will have to determine if it needs a prospective person, and, if so, find a replacement person for the billet. For example, the rule might be:

```
IF have a billet
AND have a person
AND have a prospective person
THEN one billet was filled
```

Once the knowledge base has a billet, the search for a person to fill the billet becomes more involved. A person must have a rating which is appropriate for the work center. Also, it is generally desirable to assign a person to the job he is currently in. If the current person cannot be used, then the knowledge base must search for a person who meets the billet requirements. Additionally, personnel with special qualifications should be considered first for billets in work centers where those qualifications are desired. Some of the initial thoughts for rules that determine if a person meets the billet requirements are shown below.

```
IF work center = "110"
AND person's rating = "AD"
THEN person is eligible for billet

IF work center = "120"
AND billet requirement = "AMH"
AND person's rating = "AMH"
THEN person is eligible for billet

IF work center = "120"
AND billet requirement = "AMS"
AND person's rating = "AMS"
THEN person is eligible for billet
```

```
IF work center = "12C"
AND person's rating = "AMS"
THEN person is eligible for billet

IF work center = "12A"
AND person's rating = "PR"
THEN person is eligible for billet

IF work center = "13B"
AND person's rating = "AME"
THEN person is eligible for billet

IF work center = "210"
AND person's rating = "AT"
THEN person is eligible for billet

IF work center = "220"
AND person's rating = "AE"
THEN person is eligible for billet

IF work center = "230"
AND person's rating = "AO"
THEN person is eligible for billet

IF work center = "260"
AND person's rating = "AQ"
THEN person is eligible for billet

IF work center = "020"
AND rating <= "E6"
AND person's rating = "AZ"
THEN person is eligible for billet

IF work center = "030"
AND person's rating = "AZ"
THEN person is eligible for billet

IF work center = "04A"
AND person's rating = "AZ"
THEN person is eligible for billet

IF work center = "04C"
AND person's rating = "AZ"
THEN person is eligible for billet

IF work center = "050"
AND person's rating = "AK"
THEN person is eligible for billet
```

```
IF billet requires an NEC
AND person has NEC
THEN person is eligible for billet

IF production work center
AND person is a CDI
THEN person is eligible for billet

IF work center = "110"
OR work center = "120"
OR work center = "12C
OR work center = "13A"
OR work center = "13B"
OR work center = "210"
OR work center = "220"
OR work center = "230"
OR work center = "260"
THEN production work center

IF work center = "040"
AND person is a QAR
THEN person is eligible for billet

IF work center = "230"
AND person is ordnance certified
THEN person is eligible for billet

IF work center = "310"
AND person is a plane captain
THEN person is eligible for billet

IF billet is currently filled
AND the person hasn't been assigned
AND PRD is ok
AND EAOS is ok
AND hasn't been in the job too long
THEN have a person

IF billet is currently filled
AND the person hasn't been assigned
AND PRD is ok
AND EAOS is not ok
AND person intends to reenlist
AND hasn't been in the job too long
THEN have a person
```

```
IF have a possible person
AND person meets billet requirements
AND the person hasn't been assigned
AND PRD is ok
AND EAOS is ok
THEN have a person

IF have a possible person
AND person meets billet requirements
AND the person hasn't been assigned
AND PRD is ok
AND EAOS is not ok
AND person intends to reenlist
THEN have a person
```

The next requirement is to find a prospective person. A prospective person is needed only if the current person is checking out of the squadron in thirty days or less. For example:

```
IF have a person
AND PRD > 30 days from today
AND EAOS > 30 days from today
THEN don't need a prospective person

IF have a person
AND PRD > 30 days from today
AND EAOS <= 30 days from today
AND person intends to reenlist
THEN don't need a prospective person
```

The same set of rules that applies to "have a person", as already discussed, can then be applied in searching for a prospective person.

Once all the billets have been checked, then a review of all squadron personnel will determine if all personnel are assigned. This is the last step which must be completed prior to the knowledge base reaching its goal. It involves first checking the new billet assignments to see if that person has been assigned. If the person has been assigned, then the knowledge base continues to the next person in the database. Otherwise, if the person has not been assigned, then it must find a work center to assign the person to. The rules for assigning these personnel are:

```
IF person was assigned to a work center
THEN assign to same work center

IF person's rating = "AD"
THEN assign to work center 110

IF person's rating = "AMH"
THEN assign to work center 120

IF person's rating = "AMS"
THEN assign to work center 12C

IF person's rating = "PR"
THEN assign to work center 13A

IF person's rating = "AME"
THEN assign to work center 13B

IF person's rating = "AT"
THEN assign to work center 210

IF person's rating = "AE"
THEN assign to work center 220

IF person's rating = "AO"
THEN assign to work center 230

IF person's rating = "AQ"
THEN assign to work center 280

IF person's rating = "AZ"
THEN assign to work center 020

IF person's rating = "AK"
THEN assign to work center 050
```

C. DATA DEFINITION

After examining the physical system, the need for well-defined database was clear. The major considerations were selecting the key fields and defining the data structures for both the input data and the output. The data dictionary is listed in Appendix A, and the data file structures are in Appendix B.

The data structure diagram (or Bachman diagram) is shown in Figure 3.1. It shows the relationships between records in the various data files.

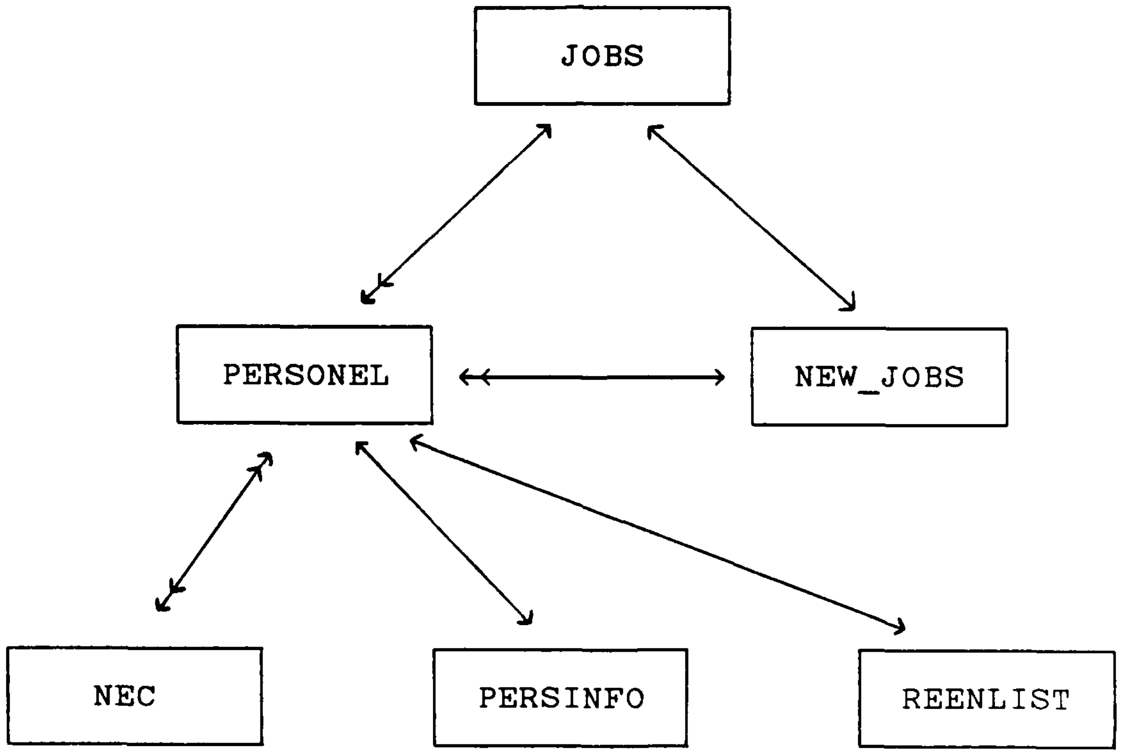

Figure 3.1 Data Structure Diagram

The input data files will be discussed first. The squadron billet file, JOBS.DBF, contains the billet number, billet description, work center, minimum paygrade acceptable for this billet, maximum paygrade acceptable for this billet, rating, the social security number of the person currently filling this billet, the social security number of a prospective person to fill the billet, the job length, if an NEC is required, and the NEC if one is required. The billet number was selected as the key field since it uniquely identifies each billet. It is also used to identify the priority of each billet. The smaller the billet number, the higher the priority, and JOBS.DBF file should be sorted in billet number sequence.

The structure of this file, JOBS.DBF, contains some of the information that was initially considered for implementation as a rule. For example, the rule:

```
IF work center = "110"
AND person's rating = "AD"
THEN person is eligible for billet
```

can be accomplished using the database and the DBPAS programs. When the information from JOBS.DBF is retrieved, it contains the work center and the paygrade and rating required for the billet. The paygrade and rating can then be passed to the DBPAS program which searches the PERSONEL.DBF file. It will then return only personnel who fill the paygrade and rating requirements, thus meeting the conclusion "person is eligible for billet". The specific implementation of DBPAS programs is discussed more fully in chapter IV.

The other sources of input data come from four personnel files. In all of the personnel files, the key field is social security number. The PERSINFO.DBF file lists a person's first, middle and last names, social security number, and rate. The PERSONEL.DBF file includes a person's social security number, rating, paygrade, date reported to the squadron, PRD, EAOS, current billet assignment, date started current billet, previous billet assignment, total days assigned to that billet, if person is qualified as a CDI, the work center where qualified as a CDI, if person is qualified as a QAR, if person is qualified as a plane captain, if person is designated as ordnance certified, if the person is a designated striker, and the rating for which he is striking. The NEC.DBF file contains a person's social security number and NEC. Since a person can have many NEC's, NEC was not included in the PERSONEL.DBF file. The REENLIST.DBF file includes a person's social security number and if he intends to reenlist at his EAOS. This is the only information which is not in the database before starting the expert system. If the information becomes necessary, the expert system prompts the user, and the result is stored in the file in case it is needed for future reference.

The output file is a list of billets, the person to be assigned to that billet, and, if necessary, a prospective person to fill the billet. To simplify the output file, NEW_JOBS.DBF, contains only the billet number, the social security number of the person assigned, and the social security number of the prospective person. Since the knowledge base goes through the JOBS.DBF file sequentially, NEW_JOBS.DBF will be listed in the same sequence as JOBS.DBF. If there are additional personnel who are not assigned after all billets are filled, then they are assigned to a work center. To simplify identification of these billets in the database, a billet number of 9999 is assigned and the work center is identified where the prospective person's social security number would otherwise be.

IV. IMPLEMENTATION

The first step taken to implement this prototype was to write database interface programs. The interfaces are written in Insight's DBPAS language and the source code is included in Appendix C. In addition, two Turbo Pascal programs are listed in Appendix D. Turbo Pascal was utilized because it could perform some specific functions that could not be accomplished using DBPAS. The next phase was writing and testing the expert system rules. The knowledge base listing is shown in Appendix E.

The flow diagrams of the expert system are shown in Figures 4.1 through 4.5. They provide an overall view of the system implementation and show how the database programs interface with the knowledge base. The flow diagrams are intended to show the high level logical flow, and do not show every possible path through the knowledge base.

A. DATABASE INTERFACES

The knowledge base uses recursion to cycle through the JOBS.DBF file. A billet counter is incremented at each level of recursion and a billet end of file

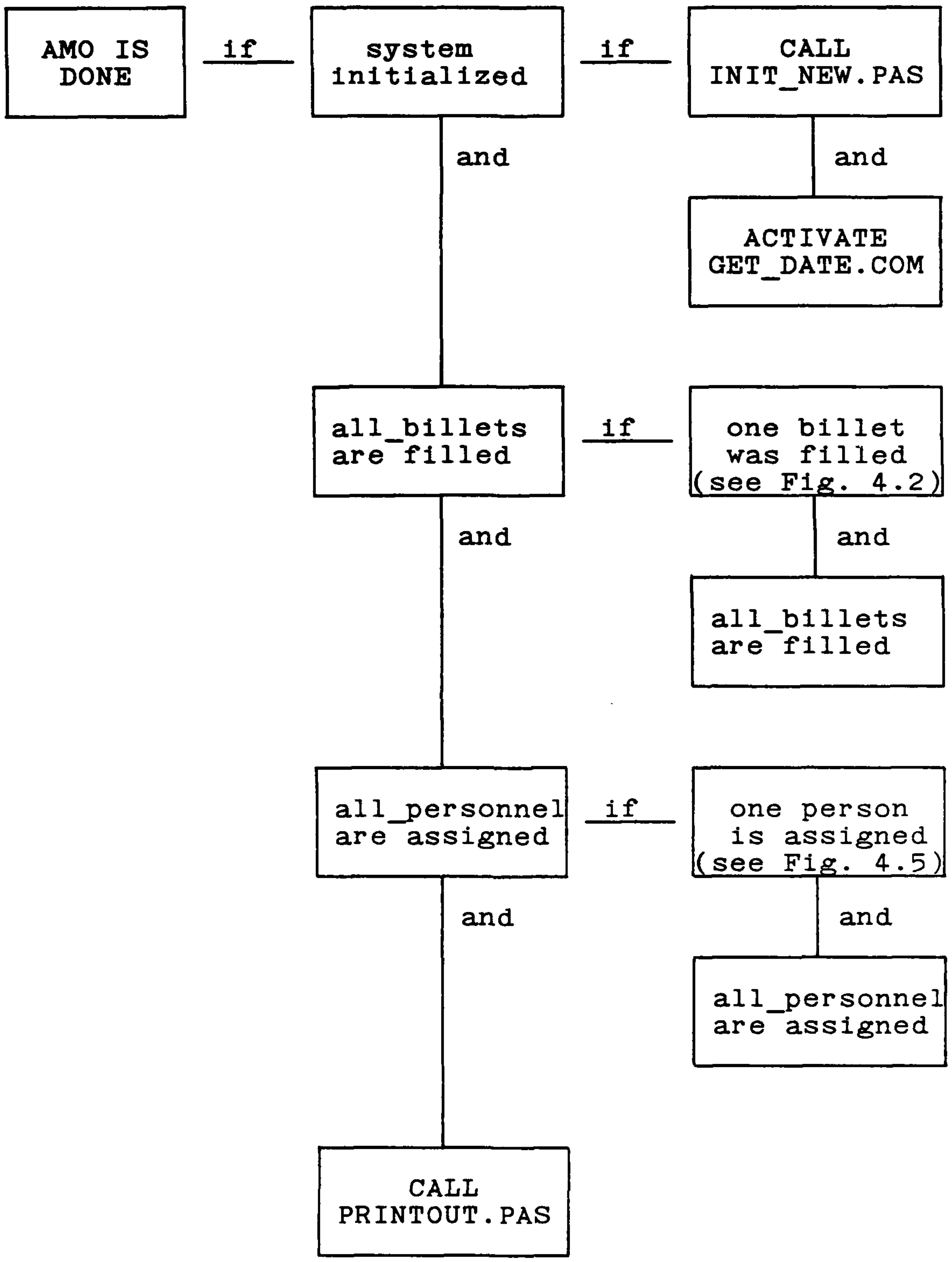

Figure 4.1 Overall Knowledge Base Flow Diagram

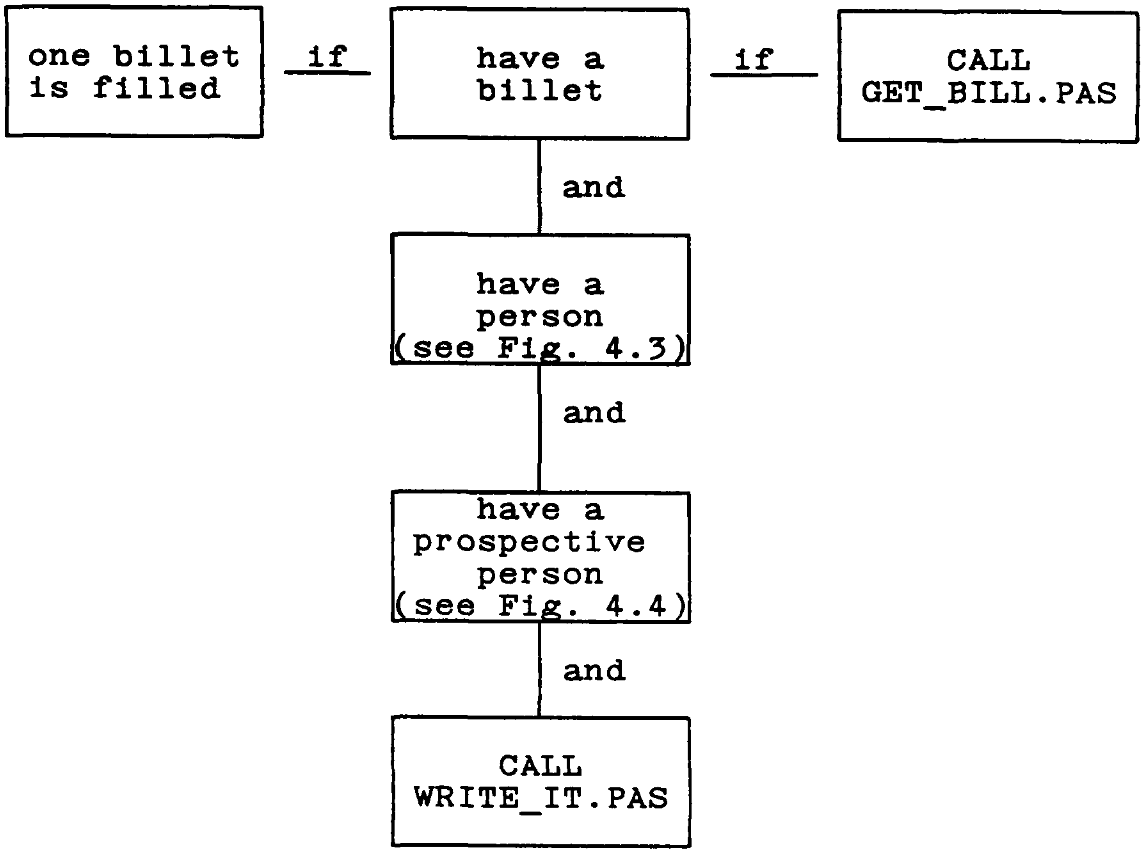

Figure 4.2 Flow Diagram For "one billet is filled"

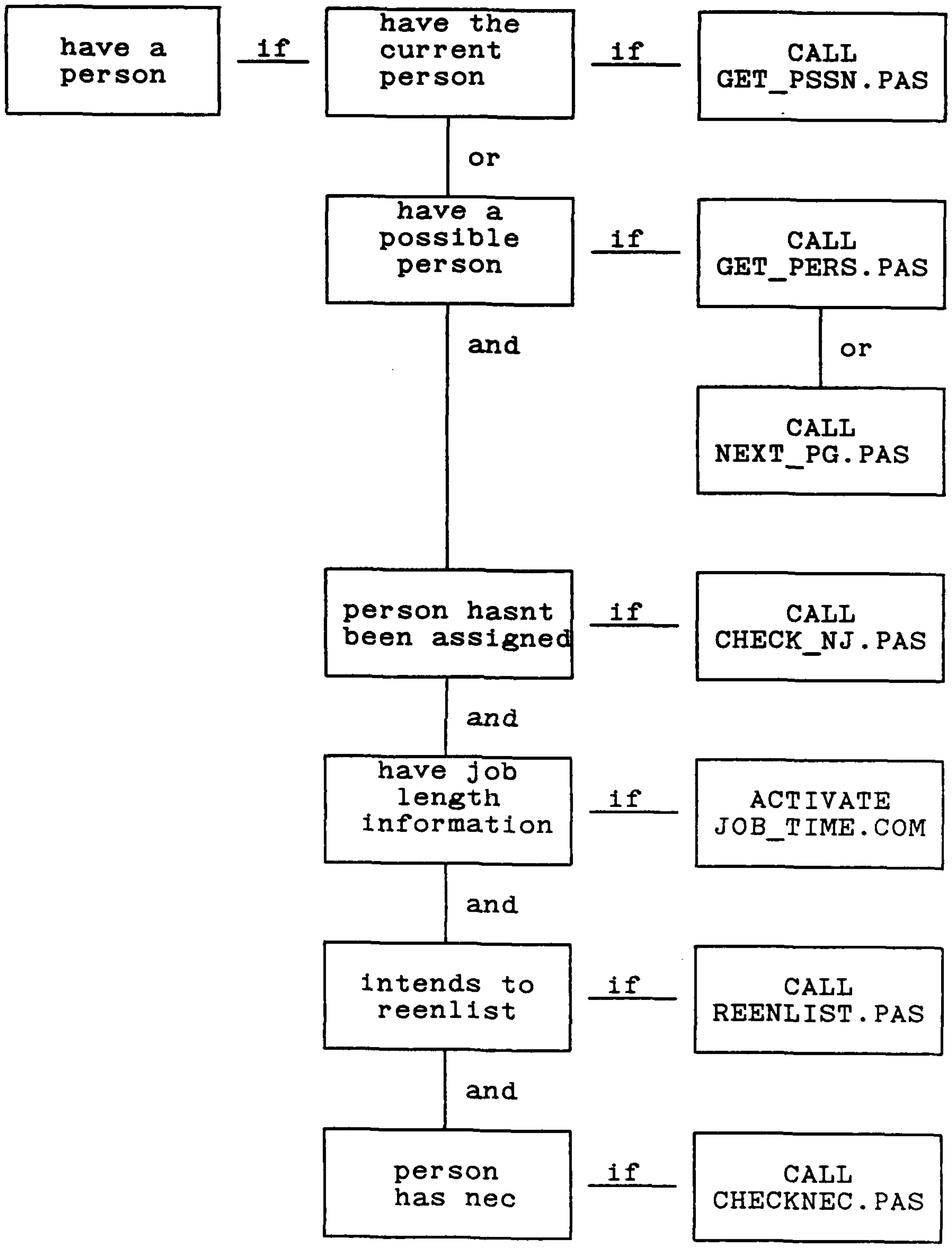

Figure 4.3 Flow Diagram For "have a person"

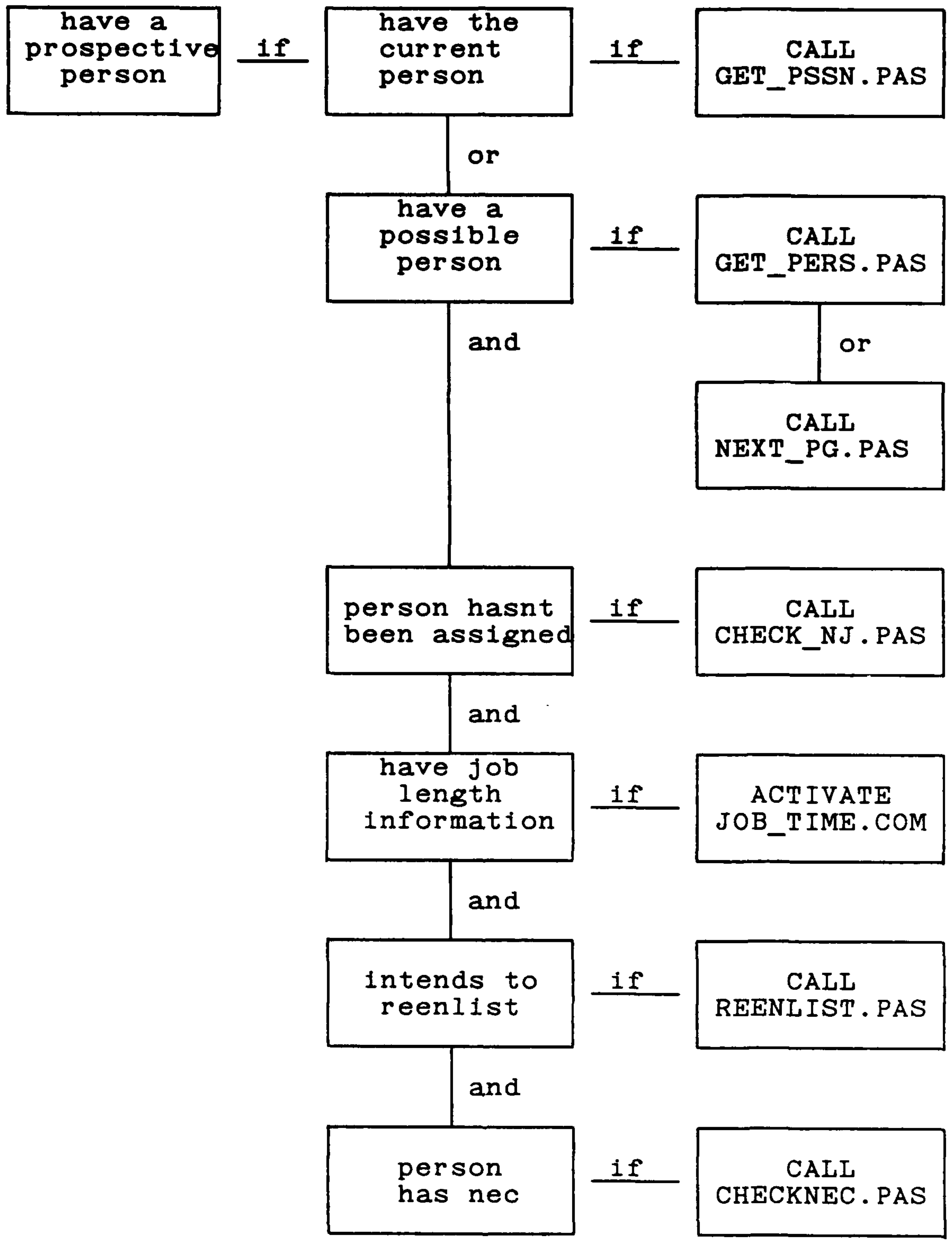

Figure 4.4 Flow Diagram For "have a prospective person"

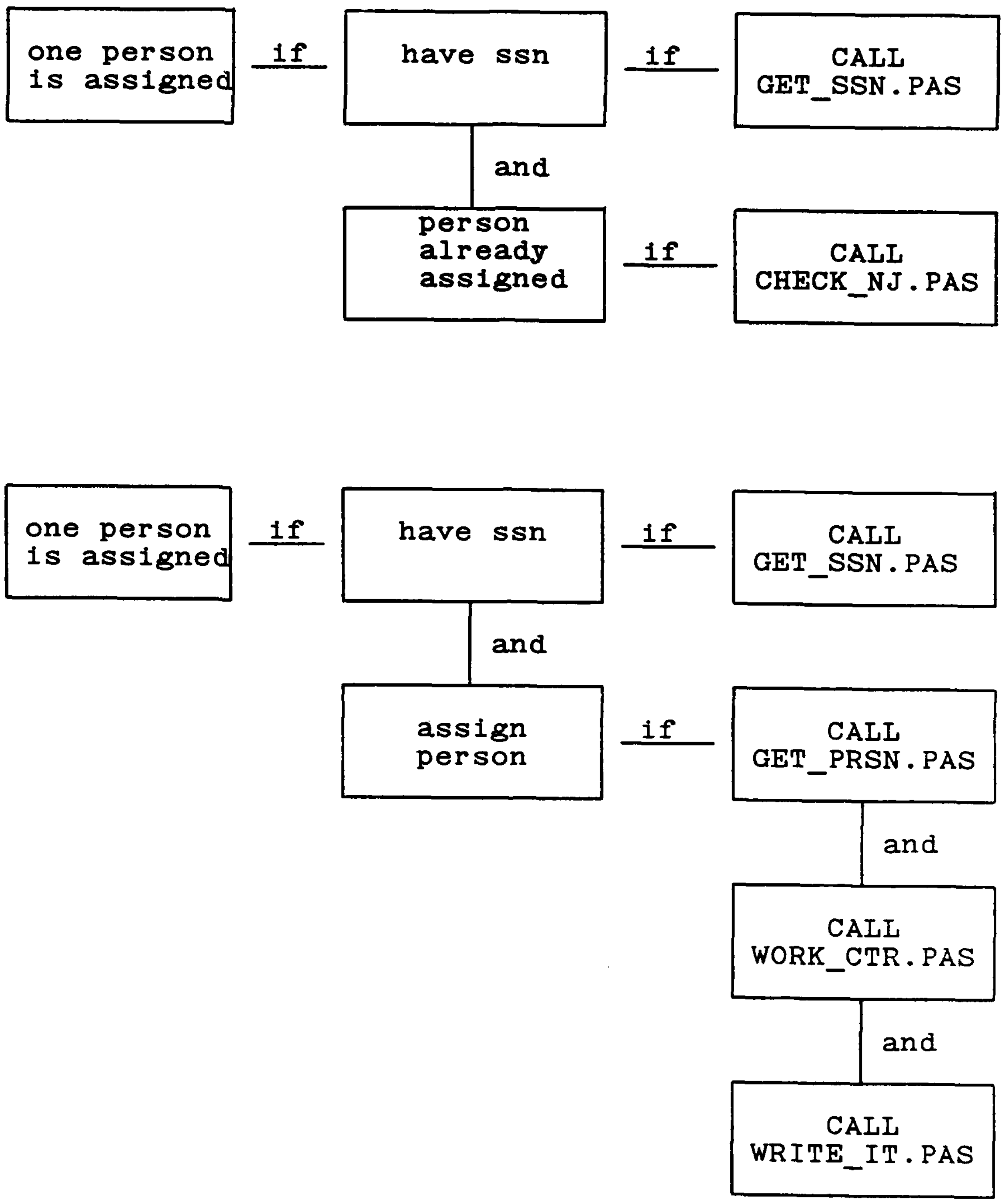

Figure 4.5 Flow Diagram For "one person is assigned"

variable identifies when all billets have been filled. The program GET_BILL.PAS receives the billet counter from the knowledge base, goes directly to that record, and returns the billet requirements and if the end of file was reached. The JOBS.DBF must be packed, i.e. not have any deleted records, since GET_BILL.PAS does not check for deleted records and will return whatever information is in that record.

Once the knowledge base has a billet, it will try to fill the billet using rules and the personnel database, PERSONEL.DBF. There are four DBPAS programs which get personnel data depending on the requirements.

The module GET_PSSN.PAS is used when the billet database, JOBS.DBF, has a person assigned as currently filling the billet or as a prospective person for the billet. The knowledge base sends a social security number to GET_PSSN.PAS which searches the personnel database until it locates that record, then it returns all the personnel data in that record to the knowledge base.

The program GET_PERS.PAS is used by the knowledge base when it does not have a person currently assigned to the billet or the person who was assigned has either been reassigned to a higher priority billet or is leaving the squadron. The knowledge base sends the record number in the personnel database where

GET_PERS.PAS should begin the search, the desired rating and the paygrade. GET_PERS.PAS returns a boolean variable identifying if a person of that rating and paygrade was found, the current record number and the person's data (if a record was found). If the knowledge base receives a person's record, it applies different rules to see if that person meets all the requirements of the billet. If the person meets all the requirements, then he is assigned to the billet; otherwise the knowledge base increments the personnel record counter, and calls GET_PERS.PAS which starts the search again. If GET_PERS.PAS did not locate a person with the right rating and paygrade, the knowledge base will call another module, NEXT_PG.PAS, which receives the current paygrade and returns the next lower pay-grade. Then the knowledge base uses rules to check if the new paygrade is greater than or equal to the minimum requirements of the billet. If it is, then the knowledge base resets the personnel record counter to the beginning of the database and calls GET_PERS.PAS to continue searching for a person with the right rating and paygrade.

The other two program which get personnel data are GET_SSN.PAS and GET_PRSN.PAS. Both are used to check if all personnel have been assigned. Once the knowl-edge base has completed all billet assignments, it uses recursion to repeatedly check the personnel database to ensure all personnel have been assigned. It begins by initializing the personnel record counter and calling GET_SSN.PAS which returns the person's social security number and a boolean variable indicating if the end of file was reached. The knowledge base sends the person's social security number to GET_PRSN.PAS which checks the NEW_JOBS.DBF to see if that social security number has been assigned to a billet. If so, the knowledge base increments the personnel record counter, and repeats the process until the end of the personnel file is reached. If the person has not been assigned to a billet, then the knowledge base calls GET_PRSN.PAS which gets the person's record from the personnel database so that the knowledge base can apply the necessary rules to assign this additional person to a work center.

During the process of assigning the additional personnel, the knowledge base applies information such as the previous billet held, the work center and the job length. When the program WORK_CTR.PAS is called, it receives a billet number, searches through the JOBS.DBF file and returns the work center and the job length of that particular billet.

The module, WRITE_IT.PAS, is used anytime the knowledge base has matched a billet to a person and a prospective person. The program receives the billet number, the person's social security number and the prospective person's social security number and appends the NEW_JOBS.DBF file.

The knowledge base calls CHECK_NJ.PAS when it has a possible person to assign to a billet and it wants to verify that the person has not been previously assigned. CHECK_NJ.PAS receives a social security number and checks every record in the NEW_JOBS.DBF file trying to match social security numbers. If it finds a match, then it returns a boolean variable of true; otherwise it returns a value of false.

The program CHECKNEC.PAS is used to determine if a person has an NEC which is required for a particular billet. CHECKNEC.PAS receives a person's social security number and the required NEC, then it searches NEC.DBF trying to match of both the social security number and NEC. If it finds a match, it returns a boolean value of true; otherwise it returns a value of false.

The module REENLIST.PAS is utilized by the knowledge base when it has a possible person to fill a billet, but the person's EAOS is less than thirty days from today's date. The module performs two functions. The first one is to query the user or the REENLIST.DBF file as to the person's reenlistment intention and to return that information to the knowledge base. The second function is to write the social security number and the person's reenlistment intention to the REENLIST.DBF file after the user has been queried. This allows possible future queries for the same social security number to be answered from the data file, rather than requesting the information again from the user. The program receives a social security number and searches the REENLIST.DBF for a match. If a match is found, then it returns the person's reenlistment intention. If no match is found, then it searches the PERSINFO.DBF to obtain the person's name and rating. The program then queries the user in the format "Does AZ2 John Adams intend to reenlist? (Y/N)". The response is appended to the REENLIST.DBF and returned to the knowledge base.

When the knowledge base has completed assigning all billets and personnel, it displays the new billet assignments using PRINTOUT.PAS. The desired format is to display the billet description, the work center, the person's rate and last name. If a prospective person is required, then the next line displays "replacement is" and the person's rate and name. PRINTOUT.PAS uses three database files, NEW_JOBS.DBF, JOBS.DBF and PERSINFO.DBF, to get the required information. The DBPAS language does not have the facilities to send the output to the printer, so the output is only displayed on the screen.

The last DBPAS program is INIT_NEW.PAS and it initializes two databases, NEW_JOBS.DBF and REENLIST.DBF. All records in both data files should be deleted prior to starting the knowledge base. The preferred method to initialize the databases is to use dBase III to delete all records and pack the databases. The dBase command "zap" will accomplish this in one step.

The module INIT_NEW.PAS checks both databases and deletes any records that exist, however it cannot pack the database. So any records that exist in NEW_JOBS.DBF or REENLIST.DBF prior to starting the knowledge will be marked for deletion, but will actually still exist in the database. Not packing the databases will cause NEW_JOBS.DBF to grow by approximately 180 records every time the knowledge base is run. This will cause slower response times when the knowledge base calls CHECK_NJ.PAS to determine if a person has already been assigned.

B. TURBO PASCAL INTERFACES

Two Turbo Pascal functions were written to perform special functions. Both programs utilize Insight 2+'s parameter passing program, ASCIIPRM.PAS, and the programs will not compile without it. The ASCIIPRM.PAS source file is a part of the Insight 2+ package and is not included in the appendices. The program GET_DATE.PAS was written to gain access to the system clock, so the system would not have to query the user for the date. It returns the current date and what the date will be thirty days from current date, since this was decided on as the planning horizon for this prototype.

The other Turbo Pascal program, JOB_TIME.PAS, is used by the knowledge base to determine if a person has exceeded the job length requirement at his current billet. This is especially critical in TAD assignments where people are rotated every ninety days. The program receives the start date, the current date and the job length from the knowledge base and returns two boolean values, (1) if the job length has been exceeded and (2) if the job length will be exceeded in thirty days. The program necessitated the use of Turbo Pascal rather than DBPAS because of the requirement to convert a character string to a number. The file PERSONEL.DBF contains the date a person started his current billet, and, when accessed by a DBPAS program, the date is returned as a character string in the form "19870215". To determine if a person has exceeded the job length, the job start date and the current date are converted to integers for year, month and day, and then actual amount of time the person has spent in the billet is calculated. This number is then compared with job length to determine if job length has been exceeded. It checks if the job length will be exceeded in the next thirty days by taking the actual time spent in the billet and adding thirty to it, and then comparing this number to job length.

C. KNOWLEDGE BASE

The knowledge base, AMO.PRL, has a single goal, AMO_IS_DONE. The goal is reached by accomplishing three tasks: all billets are filled, all personnel are assigned, and the final results are displayed. The control structures for both "all_billets are filled" and "all_personnel are assigned" are accomplished using recursion. An example of the structure is:

```
  IF one billet is filled
  AND all_billets are filled
  THEN all_billets are filled
```

The stopping condition for the recursion is:

```
  IF billet end of file
  THEN all_billets are filled
```

During implementation, the program kept getting an error message, possible infinite recursion detected, which was caused by exceeding the number of levels of recursion allowed by Insight. The rules were modified to reduce the number of recursive calls by assigning approximately twenty billets to be filled for every level of recursion.

Once the knowledge base has a billet, it must start searching through the database to find an appropriate person. The first rules deal with checking the current person in the billet to see if that person is still available to fill the billet and to check that the person has not exceeded the job length. If there is no one currently filling the billet or the person who was filling the billet is not available, then the knowledge base must start searching through the database to find someone to fill the billet. Two simplifications were made during the implementation of the prototype. The first was to assign the most senior person in paygrade to the billet, and the second was to write "none found" if the knowledge base had completed a thorough search and no one could be found to fill the billet. The knowledge base initializes the desired paygrade to the maximum allowable paygrade for that billet and searches through the database until it finds a person. If no one is found, then it desired paygrade is reduced by one paygrade and the search continues. If no one is found, it decreases the desired paygrade until it is less than the minimum allowable paygrade for that billet, and it instead of identifying a social security number, it will write "none found".

The knowledge base takes into account special qualifications in its search by only looking for personnel with the desired qualifications during its first pass through the database. For example, in a production work center which has CDIs assigned, assigning CDI's to work centers takes precedence over other considerations. The knowledge base takes this into account by searching the database for a CDI and only after it has found no CDIs available, will it restart the search for some one who is not a CDI. Although it normally assigns the person with the highest paygrade, since CDIs take precedence in production work centers, it will assign a lower paygrade person who is a CDI over a higher paygrade person who is not a CDI. For example, it will assign an E4 who is a CDI before it assigns an E5 who is not a CDI to a production work center billet. The knowledge base follows this same logic in making assignments to work centers which require special qualifications such as QAR, ordnance certified or plane captain.

V. RECOMMENDATIONS FOR FURTHER STUDY

Capturing the knowledge that assistant maintenance officers use in assigning personnel is a complex and time consuming task. This prototype utilizes only a limited part of the actual decision process. The prototype requires significant testing in real world situations and knowledge acquisition from experts to more fully represent all aspects of the process.

Once a prototype has been completed, Harmon and King [Ref. 3] state:

> An adage popular among knowledge engineers is that it is usually best to throw away the prototype. Knowledge engineering tools support rapid prototyping with a low investment of time. Thus, at this stage it is common to rethink the basic design of the knowledge base. By this we do not mean that one abandons a particular tool. We mean that the exact list of objects and attributes to be included in the system will probably change somewhat. Hierarchial relationships may need to be arranged. The exact way in which inference is handled in the heuristics may be modified as the expert and the knowledge engineer realize how the expert's knowledge and problem solving strategies can be best represented.

Based on the experience gained in building this expert system, some significant modifications to the existing prototype can be recommended. The first major change should be to expand the expert system to include all projected billet changes for the next six months. In developing the prototype system, a planning factor of thirty days ahead was considered sufficient. However, for development of a full-scale model that more closely resembles real world planning, a time frame of six months in advance is necessary.

The second major change would be to modify the overall control structure of the prototype. Rather than trying to assign a person to every billet in the database, it would be more logical to scan through the billet file and determine every billet that is not currently filled or will be vacated in a future period of time, i.e. the next six months. An output file, listing the billet number and the date the billet will be vacated, would be written to the disk. Four of the basic rules to determine if a billet will be vacated would be similar to the following:

```
IF billet is currently vacant
THEN billet will be vacated

IF current person's prd <= 6 months from today
THEN billet will be vacated

IF current person's eaos <= 6 months from today
AND person does not intend to reenlist
THEN billet will be vacated

IF job start date + job length <= 6 months from today
THEN billet will be vacated
```

The next task would be to search through the personnel data base and find all personnel who are not currently assigned to a billet, or are returning from a TAD assignment within the next six months. This would include person's who have not yet checked-in to the squadron, but appear on the EDVR as prospective gains. A file would then be written to the disk with the expected personnel gains and the expected dates.

The final task would be to match the projected vacant billets with the anticipated gains. A search algorithm could be used to arrive at a solution which offers the fewest vacancies. An additional consideration would be to add rules that would weight the billet vacancies. Higher priority billets would "cost" more to be left vacant than lower priority billets and a search algorithm would produce the least "cost" solution.

A third change to the current prototype would be to simplify the rules. Most of the current rules are long, complicated and not easy to follow. The rules could be simplified by combining several related factor into classification type rules. For example, the person's report date, PRD and EAOS could all be combined to yield a single classification that can be used by other rules. Sample rules which would fit into the current prototype might be:

```
IF report date <= today's date
AND PRD > date in one month
AND EAOS > date in one month
THEN person's tour remaining = long
```

```
IF report date <= today's date
AND PRD > date in one month
AND EAOS <= date in one month
AND person intends to reenlist
THEN person's tour remaining = long

IF report date <= today's date
AND PRD > date in one month
AND EAOS <= date in one month
AND NOT person intends to reenlist
THEN person's tour remaining = short

IF report date <= today's date
AND PRD <= date in one month
THEN person's tour remaining = short
```

This type of classification would also be useful in an expert system which was included all personnel changes within a six month time frame.

An area which is related to assigning of personnel to billets and is another major function of the assistant maintenance officer is maintenance training. The expert system could be expanded to include training requirements for particular billets and a database of the courses which personnel have successfully completed. Additionally, a database containing the course schedules could also be included, and training requirements could not only be identified, but course scheduling could also be accomplished.

Another recommendation is to perform extensive field testing of the prototype and have human "experts" point out mistakes. The mistakes would identify what knowledge was lacking, and then additional knowledge would be added to overcome its shortcomings. The prototype has not been tested extensively and cannot be considered an "expert" until comprehensive testing has been accomplished.

The knowledge base prototype accesses dBase III files for billet and personnel data. However, there is no database program to assist the user in editing, appending or deleting records in the database. The implementation of an easy-to-use database manager would be essential prior to field testing of the prototype.

VI. CONCLUSION

The Navy currently does not have any tools for automating the decision-making process for aviation maintenance personnel. Many of the criteria that are used to assign personnel in squadrons are readily captured in a heuristic knowledge base.

This project demonstrated that it is technically feasible to capture the knowledge an assistant maintenance officer uses to assign personnel in a knowledge base. Additionally, it demonstrated that a database created from the OPNAV 1000/2 and EDVR data can be integrated into an effective tool for matching authorized billets and personnel. Further work should be done to fully implement such a system, not to replace the assistant maintenance officer, but to enhance his decision making.

LIST OF REFERENCES

1. Chief of Naval Operations, OPNAVINST 4790.2D, The Naval Aviation Maintenance Program (NAMP), Volume II, p. 4-7, 1 October 1986.

2. Waterman, Donald A., A Guide to Expert Systems, p. 143, Addison-Wesley Publishing Co., 1986.

3. Harmon, Paul and King, David, Expert Systems Artificial Intelligence in Business, pp. 203-204, John Wiley and Sons, Inc., 1985.

APPENDIX A

DATA DICTIONARY

Name:	billet counter
Aliases:	file_ctr
Description:	Represents the record number of the billet which the knowledge base is currently trying to fill.
Format:	Integer, 4 digits.
Module/Files:	AMO.PRL, GET_BILL.PAS

Name:	billet description
Aliases:	job_descr, job_descrp
Description:	Describes the type of billet, i.e. supervisor, technician, etc.
Format:	Integer, 4 digits.
Module/Files:	AMO.PRL, GET_BILL.PAS, PRINTOUT.PAS, WORK_CTR.PAS, JOBS.DBF

Name:	billet eof
Aliases:	endoffile
Description:	Identifies if the end of JOBS.DBF has been reached.
Format:	Boolean, allowable values T or F.
Module/Files:	AMO.PRL, GET_BILL.PAS, CHECK_NJ.PAS

Name:	billet number
Aliases:	billet_nr
Description:	Identifies squadron billets; it also signifies the priority of the billet, lower numbers are higher priority billets.
Format:	Integer, 5 digits, numbers are in increments of 10.
Module/Files:	AMO.PRL, INIT_NEW.PAS, WRITE_IT.PAS, GET_BILL.PAS, CHECK_NJ.PAS, WORK_CTR.PAS, PRINTOUT.PAS, JOBS.DBF, NEW_JOBS.DBF

Name:	billet rating
Aliases:	rating, rating_in
Description:	Represents the rating required by that particular billet.
Format:	Alphanumeric, 3 characters, allowable values AZ, AV, AT, AE, AQ, AD, AM, AMH, AMS, AME, AO, PR, ANY.
Module/Files:	AMO.PRL, GET_BILL.PAS, GET_PERS.PAS, WORK_CTR.PAS, PRINTOUT.PAS, JOBS.DBF

Name:	cdi
Aliases:	None
Description:	Identifies if a person is qualified as a collateral duty inspector (CDI).
Format:	Boolean, allowable values T or F.
Module/Files:	AMO.PRL, GET_PERS.PAS, GET_PSSN.PAS, GET_SSN.PAS, GET_PRSN.PAS, PERSONEL.DBF

Name:	cdi for work center
Aliases:	cdi_wc
Description:	Identifies the work center where a person is qualified as a CDI.
Format:	Alphanumeric, 3 characters, allowable values 110, 120, 12C, 13A, 13B, 210, 220, 230, 260, 280.
Module/Files:	AMO.PRL, GET_PERS.PAS, GET_PSSN.PAS, GET_SSN.PAS, GET_PRSN.PAS, PERSONEL.DBF

Name:	current job
Aliases:	curr_job
Description:	Identifies the billet number of the billet a person is currently assigned to.
Format:	Integer, 4 digits.
Module/Files:	AMO.PRL, GET_PERS.PAS, GET_PSSN.PAS, GET_SSN.PAS, GET_PRSN.PAS, PERSONEL.DBF

Name:	current ssn
Aliases:	curr_ssn, curr_pers, assign_ssn
Description:	Represents the social security number of the person currently assigned to a particular billet.
Format:	Alphanumeric, 11 characters, ###-##-####.
Module/Files:	AMO.PRL, INIT_NEW.PAS, GET_BILL.PAS, GET_PSSN.PAS, CHECK_NJ.PAS, WORK_CTR.PAS, PRINTOUT.PAS, WRITE_IT.PAS, JOBS.DBF, NEW_JOBS.DBF

Name:	date plus one month
Aliases:	date30
Description:	Represents the date 30 days from today's date.
Format:	Alphanumeric, 8 characters, YYYYMMDD, for example 19870131.
Module/Files:	AMO.PRL, GET_DATE.PAS

Name:	date reported
Aliases:	date_rpt, date_reptd
Description:	Represents the date a person checked into the squadron.
Format:	Alphanumeric, 8 characters, YYYYMMDD, for example 19870131.
Module/Files:	AMO.PRL, GET_BILL.PAS, GET_PSSN.PAS, GET_SSN.PAS, GET_PRSN.PAS, PRINTOUT.PAS, PERSONEL.DBF

Name:	desired paygrade
Aliases:	paygr, paygd
Description:	Represents the paygrade of the billet which the knowledge base is currently trying to fill.
Format:	Alphanumeric, 2 characters, allowable values E1 through E9.
Module/Files:	AMO.PRL, GET_PERS.PAS, NEXT_PG.PAS

Name:	eaos
Aliases:	None
Description:	Represents a person's end of obligated active service (EAOS) date.
Format:	Alphanumeric, 8 characters, YYYYMMDD, for example 19870131.
Module/Files:	AMO.PRL, GET_PERS.PAS, GET_PSSN.PAS, GET_SSN.PAS, GET_PRSN.PAS, PRINTOUT.PAS, PERSONEL.DBF

Name:	exceeded job length
Aliases:	exceeded
Description:	Identifies if a person has been in the billet the maximum amount of time required; usually pertains to TAD billets of 90 days duration.
Format:	Boolean, allowable values T or F.
Module/Files:	AMO.PRL, JOB_TIME.PAS

Name:	first_name
Aliases:	None
Description:	Identifies a person's first name.
Format:	Alphanumeric, 15 characters.
Module/Files:	PRINTOUT.PAS, REENLIST.PAS, PERSINFO.DBF
Name:	intends to reenlist
Aliases:	intention, intent, r_intent
Description:	Identifies if a person intends to reenlist.
Format:	Boolean, allowable values T or F.
Module/Files:	AMO.PRL, REENLIST.PAS, REENLIST.DBF, INIT_NEW.PAS

Name:	job length
Aliases:	job_length, job_lngth
Description:	Identifies how long (number of days) that a job lasts, i.e. TAD billets are usually 90 days.
Format:	Integer, 4 digits.
Module/Files:	AMO.PRL, GET_BILL.PAS, WORK_CTR.PAS, PRINTOUT.PAS, JOB_TIME.PAS, JOBS.DBF

Name:	last_name
Aliases:	None
Description:	Identifies a person's last name.
Format:	Alphanumeric, 15 characters.
Module/Files:	PRINTOUT.PAS, REENLIST.PAS, PERSINFO.DBF

Name:	maximum paygrade
Aliases:	maxpaygr, max_paygr
Description:	Represents the highest allowable paygrade for a particular billet.
Format:	Alphanumeric, 2 characters, allowable values E1 through E9.
Module/Files:	AMO.PRL, GET_BILL.PAS, WORK_CTR.PAS, PRINTOUT.PAS, JOBS.DBF

Name:	middle_nam
Aliases:	None
Description:	Identifies a person's middle name.
Format:	Alphanumeric, 15 characters.
Module/Files:	PRINTOUT.PAS, REENLIST.PAS, PERSINFO.DBF

Name:	minimum paygrade
Aliases:	minpaygr, min_paygr
Description:	Represents the lowest allowable paygrade for a particular billet.
Format:	Alphanumeric, 2 characters, allowable values E1 through E9.
Module/Files:	AMO.PRL, GET_BILL.PAS, WORK_CTR.PAS, PRINTOUT.PAS, JOBS.DBF

Name:	nec code
Aliases:	nec_code, nec
Description:	Represents the Navy Enlisted Classification (NEC) code which identifies a person's particular skill.
Format:	Integer, 4 digits.
Module/Files:	AMO.PRL, GET_BILL.PAS, WORK_CTR.PAS, PRINTOUT.PAS, CHECKNEC.PAS, NEC.DBF, JOBS.DBF

Name:	nec found
Aliases:	nec_fnd
Description:	Identifies if the NEC was found in the NEC.DBF file.
Format:	Boolean, allowable values T or F.
Module/Files:	AMO.PRL, CHECKNEC.PAS

Name:	nec required
Aliases:	nec_reqd
Description:	Identifies if the billet requires a
person	with a specific NEC.
Format:	Boolean, allowable values T or F.
Module/Files:	AMO.PRL, GET_BILL.PAS, WORK_CTR.PAS, PRINTOUT.PAS, JOBS.DBF

Name:	ordnance certified
Aliases:	ord_cert
Description:	Identifies if a person is qualified to handle ordnance.
Format:	Boolean, allowable values T or F.
Module/Files:	AMO.PRL, GET_PERS.PAS, GET_PSSN.PAS, GET_SSN.PAS, GET_PRSN.PAS, PERSONEL.DBF

Name:	paygrade
Aliases:	None
Description:	Represents the person's paygrade.
Format:	Alphanumeric, 2 characters, allowable values E1 through E9.
Module/Files:	AMO.PRL, GET_PERS.PAS, GET_PSSN.PAS, GET_SSN.PAS, GET_PRSN.PAS, PERSONEL.DBF

Name:	person assigned
Aliases:	None
Description:	Identifies if a person has already been assigned to a billet.
Format:	Boolean, allowable values T or F.
Module/Files:	AMO.PRL

Name:	personnel counter
Aliases:	recrd_nr, new_recn
Description:	Represents the record number of the person that the knowledge base is currently trying to match with a billet.
Format:	Integer, 4 digits.
Module/Files:	AMO.PRL, GET_PERS.PAS, GET_PSSN.PAS, GET_SSN.PAS, GET_PRSN.PAS

Name:	personnel eof
Aliases:	endoffile
Description:	Identifies if the end of file has been reached in the PERSONEL.DBF.
Format:	Boolean, allowable values T or F.
Module/Files:	AMO.PRL, GET_SSN.PAS

Name:	persons rating
Aliases:	rating
Description:	Represents a person's rating.
Format:	Alphanumeric, 3 characters, allowable values AZ, AV, AT, AE, AQ, AD, AM, AMH, AMS, AME, AO, PR, NON.
Module/Files:	AMO.PRL, GET_PERS.PAS, GET_PSSN.PAS, GET_SSN.PAS, GET_PRSN.PAS, PERSONEL.DBF

Name:	plane captain
Aliases:	plane_capt, pc
Description:	Identifies if a person is qualified as a plane captain.
Format:	Boolean, allowable values T or F.
Module/Files:	AMO.PRL, GET_PERS.PAS, GET_PSSN.PAS, GET_SSN.PAS, GET_PRSN.PAS, PERSONEL.DBF

Name:	prd
Aliases:	None
Description:	Represents a person's projected rotation date (PRD).
Format:	Alphanumeric, 8 characters, YYYYMMDD, for example 19870131.
Module/Files:	AMO.PRL, GET_PERS.PAS, GET_PSSN.PAS, GET_SSN.PAS, GET_PRSN.PAS, PRINTOUT.PAS, PERSONEL.DBF

Name:	previous job
Aliases:	prev_job
Description:	Identifies the billet number of the billet a person was previously assigned to before his currently assignment.
Format:	Integer, 4 digits.
Module/Files:	AMO.PRL, GET_PERS.PAS, GET_PSSN.PAS, GET_SSN.PAS, GET_PRSN.PAS, PERSONEL.DBF

Name:	prospective ssn
Aliases:	pros_ssn, pros_pers, future_ssn
Description:	Represents the social security number of the person who is going to be assigned to the billet within the next 30 days.
Format:	Alphanumeric, 11 characters, ###-##-####.
Module/Files:	AMO.PRL, INIT_NEW.PAS, GET_BILL.PAS, WRITE_IT.PAS, WORK_CTR.PAS, CHECK_NJ.PAS, PRINTOUT.PAS, JOBS.DBF, NEW_JOBS.DBF

Name:	qar
Aliases:	None
Description:	Identifies if a person is qualified as a quality assurance representative.
Format:	Boolean, allowable values T or F.
Module/Files:	AMO.PRL, GET_PERS.PAS, GET_PSSN.PAS, GET_SSN.PAS, GET_PRSN.PAS, PERSONEL.DBF

Name:	rate
Aliases:	None
Description:	Represents a person's rate, for example: AZ3, ATAA, AMCS, etc.
Format:	Alphanumeric, 5 characters.
Module/Files:	PRINTOUT.PAS, REENLIST.PAS, PERSINFO.DBF

Name:	social security number
Aliases:	ssn
Description:	Represents the person's social security number.
Format:	Alphanumeric, 11 characters, ###-##-####.
Module/Files:	AMO.PRL, GET_PSSN.PAS, CHECK_NJ.PAS, GET_SSN.PAS, GET_PRSN.PAS, PRINTOUT.PAS, REENLIST.PAS, CHECKNEC.PAS, INIT_NEW.PAS, REENLIST.DBF, NEC.DBF, PERSINFO.DBF

Name:	ssn was found
Aliases:	ssn_fnd
Description:	Identifies if a ssn was found while searching through the PERSONEL.DBF.
Format:	Boolean, allowable values T or F.
Module/Files:	AMO.PRL, GET_PERS.PAS, GET_PSSN.PAS, GET_PRSN.PAS

Name:	start date
Aliases:	start_dt, start_date
Description:	Represents the date a person started his current billet assignment.
Format:	Alphanumeric, 8 characters, YYYYMMDD, for example 19870131.
Module/Files:	AMO.PRL, GET_PERS.PAS, GET_PSSN.PAS, GET_SSN.PAS, GET_PRSN.PAS, JOB_TIME.PAS, PERSONEL.DBF

Name:	striker
Aliases:	None
Description:	Identifies if a person is striking for a particular rating; applies to personnel who are E1 through E3 and do not have a designated rating.
Format:	Boolean, allowable values T or F.
Module/Files:	AMO.PRL, GET_PERS.PAS, GET_PSSN.PAS, GET_SSN.PAS, GET_PRSN.PAS, PERSONEL.DBF

Name:	striker rating
Aliases:	str_rating, str_rate
Description:	Represents the rating a person is striking for.
Format:	Alphanumeric, 3 characters, allowable values AZ, AT, AE, AQ, AD, AMH, AMS, AME, AO, PR.
Module/Files:	AMO.PRL, GET_PERS.PAS, GET_PSSN.PAS, GET_SSN.PAS, GET_PRSN.PAS, PERSONEL.DBF

Name:	todays date
Aliases:	today_date, date
Description:	Represents today's date which is obtained from the system clock.
Format:	Alphanumeric, 8 characters, YYYYMMDD, for example 19870131.
Module/Files:	AMO.PRL, GET_DATE.PAS, JOB_TIME.PAS

Name:	total days in previous job
Aliases:	total_days, tot_days
Description:	Represents the total number of days a person was assigned to the job he held prior to his current job.
Format:	Integer, 4 digits.
Module/Files:	AMO.PRL, GET_PERS.PAS, GET_PSSN.PAS, GET_SSN.PAS, GET_PRSN.PAS, PERSONEL.DBF

Name:	total tad time
Aliases:	tad_time, tot_tad
Description:	Represents the total number of days a person has been assigned TAD from the squadron.
Format:	Integer, 4 digits.
Module/Files:	AMO.PRL, GET_PERS.PAS, GET_PSSN.PAS, GET_SSN.PAS, GET_PRSN.PAS, PERSONEL.DBF

Name:	work center
Aliases:	work_cntr, workcenter
Description:	Represents the work center that corresponds to the billet.
Format:	Alphanumeric, 3 characters, allowable values 020, 030, 040, 04A, 04C, 050, 100, 110, 120, 12C, 13A, 13B, 140, 200, 210, 220, 230, 260, 280, 300, 310, 320, 1LT, CCU, CMC, IMA, SAF, SUP, TNG, TAD.
Module/Files:	AMO.PRL, GET_BILL.PAS, WORK_CTR.PAS, PRINTOUT.PAS, JOBS.DBF

APPENDIX B

DATABASE STRUCTURE

The structure of the six database files which are accessed by the expert system are listed below.

Structure for:	JOBS.DBF
Aliases:	billet, jobsfile
Modules:	GET_BILL.PAS, WORK_CTR.PAS, PRINTOUT.PAS

Field name	Type	Width
BILLET_NR	Numeric	4
JOB_DESCRP	Character	20
WORKCENTER	Character	3
MIN_PAYGR	Character	2
MAX_PAYGR	Character	2
RATING	Character	3
CURR_PERS	Character	11
PROS_PERS	Character	11
JOB_LENGTH	Numeric	4
NEC_REQD	Logical	1
NEC	Numeric	4

Structure for:	PERSINFO.DBF
Aliases:	personnel
Modules:	REENLIST.PAS, PRINTOUT.PAS

Field name	Type	Width
LAST_NAME	Character	15
FIRST_NAME	Character	15
MIDDLE_NAM	Character	15
SSN	Character	11
RATE	Character	5

Structure for: PERSONNEL.DBF
Aliases: persfile
Modules: GET_SSN.PAS, GET_PERS.SSN, GET_PSSN.PAS
GET_PRSN.PAS

Field name	Type	Width
SSN	Character	11
RATING	Character	3
PAYGRADE	Character	2
DATE_REPTD	Date	8
PRD	Date	8
EAOS	Date	8
CURR_JOB	Numeric	4
START_DATE	Date	8
PREV_JOB	Numeric	4
TOT_DAYS	Numeric	4
CDI	Logical	1
CDI_WC	Character	3
QAR	Logical	1
PLANE_CAPT	Logical	1
ORD_CERT	Logical	1
STRIKER	Logical	1
STR_RATING	Character	3
TOT_TAD	Numeric	4

Structure for: REENLIST.DBF
Aliases: re_enlist
Modules: REENLIST.PAS, INIT_NEW.PAS

Field name	Type	Width
SSN	Character	11
INTENT	Logical	1

Structure for: NEC.DBF
Aliases: necfile
Modules: CHECKNEC.PAS

Field name	Type	Width
SSN	Character	11
NEC	Numeric	4

```
Structure for: NEW_JOBS.DBF
Aliases:       assign, newjobs
Modules:       INIT_NEW.PAS, CHECK_NJ.PAS,
               PRINTOUT.PAS, WRITE_IT.PAS
```

Field name	Type	Width
BILLET_NR	Numeric	3
CURR_SSN	Character	11
PROS_SSN	Character	11

APPENDIX C

DBPAS PROGRAMS

```
program CHECK_NEC(RECEIVE ssn     : string(11);
                          nec     : integer;
                   RETURN nec_fnd : boolean);

var
    necfile : record
       ssn  : string(11);
       nec  : real;
    end;

begin
     open( necfile, 'nec');
     goto( 1, necfile);
     while NOT EOF(necfile) AND ((necfile.ssn <> ssn)
        OR (ROUND(necfile.nec) <> nec))  do
        begin
           NEXT(necfile);
        end;
     nec_fnd := NOT EOF(necfile);
     close(necfile);
end;
```

```
program CHECK_NJ(RECEIVE ssn      : string(11);
                 RETURN  assigned : boolean);

var
    assign : record
       billet_nr : real;
       curr_ssn  : string(11);
       pros_ssn  : string(11);
    end;

begin
   open(assign, 'new_jobs');
   goto(1, assign);
   while NOT EOF(assign) AND
        ((assign.deleted = true) OR
        ((assign.curr_ssn <> ssn) AND
        (assign.pros_ssn <> ssn)))  do
      begin
         NEXT(assign);
      end;
   assigned := NOT EOF(assign);
   close(assign);
end;
```

```
program GET_BILL(RECEIVE file_ctr   : integer;
                 RETURN  billet_nr  : integer;
                         job_descr  : string(20);
                         work_cntr  : string(3);
                         minpaygr   : string(2);
                         maxpaygr   : string(2);
                         rating     : string(3);
                         curr_pers  : string(11);
                         pros_pers  : string(11);
                         job_lngth  : integer;
                         nec_reqd   : boolean;
                         nec_code   : integer;
                         endoffile  : boolean);

var
    billet : record
       billet_nr   : real;
       job_descrp  : string(20);
       workcenter  : string(3);
       min_paygr   : string(2);
       max_paygr   : string(2);
       rating      : string(3);
       curr_pers   : string(11);
       pros_pers   : string(11);
       job_length  : real;
       nec_reqd    : boolean;
       nec         : real;
    end;

begin
   open( billet, 'jobs');
   goto(file_ctr, billet);
   if EOF(billet) = true
      then

         begin
            billet_nr := 0;
            job_descr := '                    ';
            work_cntr := '   ';
            minpaygr := '  ';
            maxpaygr := '  ';
            rating := '   ';
            curr_pers := '           ';
            pros_pers := '           ';
            job_lngth := 0;
            nec_reqd := false;
            nec_code := 0;
         end

      else
```

```
         begin
            billet_nr := ROUND(billet.billet_nr);
            job_descr := billet.job_descrp;
            work_cntr := billet.workcenter;
            minpaygr := billet.min_paygr;
            maxpaygr := billet.max_paygr;
            rating := billet.rating;
            curr_pers := billet.curr_pers;
            pros_pers := billet.pros_pers;
            job_lngth := ROUND(billet.job_length);
            nec_reqd := billet.nec_reqd;
            nec_code := ROUND(billet.nec);
         end;

   endoffile := EOF(billet);
   close(billet);
end;
```

```
program GET_PERS(RECEIVE   recrd_nr : integer;
                           rating_in: string(3);
                           paygr    : string(2);
                 RETURN    ssn_fnd  : boolean;
                           pros_ssn : string(11);
                           rating   : string(3);
                           paygrade : string(3);
                           date_rpt : string(8);
                           prd      : string(8);
                           eaos     : string(8);
                           curr_job : integer;
                           start_dt : string(8);
                           prev_job : integer;
                           tot_days : integer;
                           cdi      : boolean;
                           cdi_wc   : string(3);
                           qar      : boolean;
                           pc       : boolean;
                           ord_cert : boolean;
                           striker  : boolean;
                           str_rate : string(3);
                           tad_time : integer;
                           new_recn : integer);

var
    persfile : record
       ssn         : string(11);
       rating      : string(3);
       paygrade    : string(2);
       date_reptd  : string(8);
       prd         : string(8);
       eaos        : string(8);
       curr_job    : real;
       start_date  : string(8);
       prev_job    : real;
       total_days  : real;
       cdi         : boolean;
       cdi_wc      : string(3);
       qar         : boolean;
       plane_capt  : boolean;
       ord_cert    : boolean;
       striker     : boolean;
       str_rating  : string(3);
       tot_tad     : real;
    end;

begin
     open( persfile, 'personel');
     goto( recrd_nr, persfile);
     if rating_in = 'ANY'
        then
```

```
         begin
            while NOT EOF(persfile) AND
            (persfile.paygrade <> paygr)  do
                begin
                   NEXT(persfile);
                end;
         end

      else

         begin
            while NOT EOF(persfile) AND
            ((persfile.paygrade <> paygr) OR
            (persfile.rating <> rating_in))  do
               begin
                  NEXT(persfile);
               end;
         end;

   if EOF(persfile)
      then

         begin
            ssn_fnd := false;
            pros_ssn := '           ';
            rating := '   ';
            paygrade := '   ';
            date_rpt := '        ';
            prd := '        ';
            eaos := '        ';
            curr_job := Ø;
            start_dt := '        ';
            prev_job := Ø;
            tot_days := Ø;
            cdi := false;
            cdi_wc := '   ';
            qar := false;
            pc := false;
            ord_cert := false;
            striker := false;
            str_rate := '   ';
            tad_time := Ø;
            new_recn := SIZE(persfile) + 1;
         end

      else
```

```
          begin
             ssn_fnd := true;
             pros_ssn := persfile.ssn;
             rating := persfile.rating;
             paygrade := persfile.paygrade;
             date_rpt := persfile.date_reptd;
             prd := persfile.prd;
             eaos := persfile.eaos;
             curr_job := ROUND(persfile.curr_job);
             start_dt := persfile.start_date;
             prev_job := ROUND(persfile.prev_job);
             tot_days := ROUND(persfile.total_days);
             cdi := persfile.cdi;
             cdi_wc := persfile.cdi_wc;
             qar := persfile.qar;
             pc := persfile.qar;
             ord_cert := persfile.ord_cert;
             striker := persfile.striker:
             str_rate := persfile.str_rating;
             tad_time := ROUND(persfile.tot_tad);
             new_recn := RECNO(persfile)
          end;

      close(persfile);
end;
```

```
program GET_PRSN(RECEIVE recrd_nr  : integer;
                 RETURN  ssn_fnd   : boolean;
                         ssn       : string(11);
                         rating    : string(3);
                         paygrade  : string(3);
                         date_rpt  : string(8);
                         prd       : string(8);
                         eaos      : string(8);
                         curr_job  : integer;
                         start_dt  : string(8);
                         prev_job  : integer;
                         tot_days  : integer;
                         cdi       : boolean;
                         cdi_wc    : string(3);
                         qar       : boolean;
                         pc        : boolean;
                         ord_cert  : boolean;
                         striker   : boolean;
                         str_rate  : string(3);
                         tad_time  : integer);
var
    persfile : record
       ssn         : string(11);
       rating      : string(3);
       paygrade    : string(2);
       date_reptd  : string(8);
       prd         : string(8);
       eaos        : string(8);
       curr_job    : real;
       start_date  : string(8);
       prev_job    : real;
       total_days  : real;
       cdi         : boolean;
       cdi_wc      : string(3);
       qar         : boolean;
       plane_capt  : boolean;
       ord_cert    : boolean;
       striker     : boolean;
       str_rating  : string(3);
       tot_tad     : real;
    end;

begin

     open( persfile, 'personel');
     goto( recrd_nr, persfile);

     if EOF(persfile)
        then
```

```
            begin
               ssn_fnd := false;
               ssn := '           ';
               rating := '   ';
               paygrade := '   ';
               date_rpt := '        ';
               prd := '        ';
               eaos := '        ';
               curr_job := 0;
               start_dt := '        ';
               prev_job := 0;
               tot_days := 0;
               cdi := false;
               cdi_wc := '   ';
               qar := false;
               pc := false;
               ord_cert := false;
               striker := false;
               str_rate := '   ';
               tad_time := 0;
            end

         else

            begin
               ssn_fnd := true;
               ssn := persfile.ssn;
               rating := persfile.rating;
               paygrade := persfile.paygrade;
               date_rpt := persfile.date_reptd;
               prd := persfile.prd;
               eaos := persfile.eaos;
               curr_job := ROUND(persfile.curr_job);
               start_dt := persfile.start_date;
               prev_job := ROUND(persfile.prev_job);
               tot_days := ROUND(persfile.total_days);
               cdi := persfile.cdi;
               cdi_wc := persfile.cdi_wc;
               qar := persfile.qar;
               pc := persfile.qar;
               ord_cert := persfile.ord_cert;
               striker := persfile.striker;
               str_rate := persfile.str_rating;
               tad_time := ROUND(persfile.tot_tad);
            end;

      close(persfile);
end;
```

```
program GET_PSSN(RECEIVE curr_ssn  : string(11);
                 RETURN  ssn_fnd   : boolean;
                         rating    : string(3);
                         paygrade  : string(3);
                         date_rpt  : string(8);
                         prd       : string(8);
                         eaos      : string(8);
                         curr_job  : integer;
                         start_dt  : string(8);
                         prev_job  : integer;
                         tot_days  : integer;
                         cdi       : boolean;
                         cdi_wc    : string(3);
                         qar       : boolean;
                         pc        : boolean;
                         ord_cert  : boolean;
                         striker   : boolean;
                         str_rate  : string(3);
                         tad_time  : integer;
                         new_recn  : integer);

var
    persfile : record
       ssn         : string(11);
       rating      : string(3);
       paygrade    : string(2);
       date_reptd  : string(8);
       prd         : string(8);
       eaos        : string(8);
       curr_job    : real;
       start_date  : string(8);
       prev_job    : real;
       total_days  : real;
       cdi         : boolean;
       cdi_wc      : string(3);
       qar         : boolean;
       plane_capt  : boolean;
       ord_cert    : boolean;
       striker     : boolean;
       str_rating  : string(3);
       tot_tad     : real;
    end;

begin
     open( persfile, 'personel');
     goto( 1, persfile);
     while NOT EOF(persfile) AND
             (curr_ssn <> persfile.ssn) do
        begin
           NEXT(persfile);
        end;
    if EOF(persfile)
      then
        begin
          ssn_fnd := false;
          rating := '   ';
          paygrade := '   ';
          date_rpt := '        ';
          prd := '        ';
          eaos := '        ';
          curr_job := Ø;
          start_dt := '        ';
          prev_job := Ø;
          tot_days := Ø;
          cdi := false;
          cdi_wc := '   ';
          qar := false;
          pc := false;
          ord_cert := false;
          striker := false;
          str_rate := '   ';
          tad_time := Ø;
          new_recn := SIZE(persfile) + 1;
        end

    else

        begin
          ssn_fnd := true;
          rating := persfile.rating;
          paygrade := persfile.paygrade;
          date_rpt := persfile.date_reptd;
          prd := persfile.prd;
          eaos := persfile.eaos;
          curr_job := ROUND(persfile.curr_job);
          start_dt := persfile.start_date;
          prev_job := ROUND(persfile.prev_job);
          tot_days := ROUND(persfile.total_days);
          cdi := persfile.cdi;
          cdi_wc := persfile.cdi_wc;
          qar := persfile.qar;
          pc := persfile.qar;
          ord_cert := persfile.ord_cert;
          striker := persfile.striker;
          str_rate := persfile.str_rating;
          tad_time := ROUND(persfile.tot_tad);
          new_recn := RECNO(persfile);
        end;
    close(persfile);
end;
```

```
program GET_SSN(RECEIVE recrd_nr : integer;
                RETURN  ssn      : string(11);
                        endoffile: boolean);

var
    persfile : record
       ssn         : string(11);
       rating      : string(3);
       paygrade    : string(2);
       date_reptd  : string(8);
       prd         : string(8);
       eaos        : string(8);
       curr_job    : real;
       start_date  : string(8);
       prev_job    : real;
       total_days  : real;
       cdi         : boolean;
       cdi_wc      : string(3);
       qar         : boolean;
       plane_capt  : boolean;
       ord_cert    : boolean;
       striker     : boolean;
       str_rating  : string(3);
       tot_tad     : real;
    end;

begin
     open( persfile, 'personel');
     goto( recrd_nr, persfile);
     if EOF(persfile)
        then
           begin
              ssn := '           ';
           end

        else

           begin
              ssn := persfile.ssn;
           end;

     endoffile := EOF(persfile);
     close(persfile);
end;
```

```
program INIT_NEW;

var
    assign : record
       billet_nr : real;
       curr_ssn  : string(11);
       pros_ssn  : string(11);
    end;

    re_enlist : record
       ssn  : string(11);
       intent : boolean;
    end;

begin
   open(assign, 'new_jobs');
   goto(1, assign);
   while NOT EOF(assign) do
      begin
         if assign.deleted = false
            then
               begin
                  assign.deleted := true;
                  replace(assign);
               end;
         NEXT(assign);
      end;
   close(assign);

   open(re_enlist, 'reenlist');
   goto(1, re_enlist);
   while NOT EOF(re_enlist) do
      begin

         if re_enlist.deleted = false
            then
               begin
                  re_enlist.deleted := true;
                  replace(re_enlist);
               end;
         NEXT(re_enlist);

      end;
   close(re_enlist);
end;
```

```
program NEXT_PG(RECEIVE paygr : string(2);
                RETURN  paygd : string(2));

begin
   if paygr = 'E9'
      then
         paygd := 'E8';

   if paygr = 'E8'
      then
         paygd := 'E7';

   if paygr = 'E7'
      then
         paygd := 'E6';

   if paygr = 'E6'
      then
         paygd := 'E5';

   if paygr = 'E5'
      then
         paygd := 'E4';

   if paygr = 'E4'
      then
         paygd := 'E3';

   if paygr = 'E3'
      then
         paygd := 'E2';

   if paygr = 'E2'
      then
         paygd := 'E1';

   if paygr = 'E1'
      then
         paygd := 'E0';

end;
```

```
program PRINT_OUT;

var
    newjobs : record
       billet_nr : real;
       curr_ssn  : string(11);
       pros_ssn  : string(11);
    end;

    personnel : record
       last_name   : string(15);
       first_name  : string(15);
       middle_nam  : string(15);
       ssn         : string(11);
       rate        : string(5);
    end;

    jobsfile : record
       billet_nr   : real;
       job_descrp  : string(20);
       workcenter  : string(3);
       min_paygr   : string(2);
       max_paygr   : string(2);
       rating      : string(3);
       curr_pers   : string(11);
       pros_pers   : string(11);
       job_length  : real;
       nec_reqd    : boolean;
       nec         : real;
    end;

    dx_nx : string(2);

begin
   open(newjobs, 'new_jobs');
   open(personnel, 'persinfo');
   open(jobsfile, 'jobs');
   goto(1, newjobs);
   while NOT EOF(newjobs) do
      begin
         if (newjobs.deleted = false) AND
            (newjobs.billet_nr <> 9999.0) then
            begin
               goto(1, jobsfile);
               while NOT EOF(jobsfile) AND
                  (newjobs.billet_nr <>
                   jobsfile.billet_nr) do
                   begin
                      NEXT(jobsfile);
                   end;
      goto(1, personnel);

      while NOT EOF(personnel)   AND
            (newjobs.curr_ssn <>
                  personnel.ssn) AND
            (newjobs.curr_ssn <>
                  'none found') do
         begin
            NEXT(personnel);
         end;

      if (newjobs.curr_ssn = 'none found')
         then
         WRITELN(jobsfile.job_descrp,'   ',
                 jobsfile.workcenter,
                 '   no one found')
         else
         WRITELN(jobsfile.job_descrp,'   ',
                 jobsfile.workcenter,
                 '   ',personnel.rate,'   ',
                 personnel.last_name);

      if (newjobs.pros_ssn > '000-00-0000') OR
         (newjobs.pros_ssn = 'none found')
         then
            begin
               goto(1, personnel);
               while NOT EOF(personnel) AND
                     (newjobs.pros_ssn <>
                      personnel.ssn) AND
                     (newjobs.pros_ssn <>
                     'none found') do
                  begin
                     NEXT(personnel);
                  end;

               if (newjobs.pros_ssn =
                     'none found') then
                  WRITELN('     replacement
                   needed, but no one found')
                  else
                  WRITELN('     replacement
                    is    ',personnel.rate,
                    ' ',personnel.last_name);
            end;
      WRITELN;
   end;
```

```
            if (newjobs.deleted = false) AND
               (newjobs.billet_nr = 9999.0)  then
               begin

                  goto(1, personnel);
                  while NOT EOF(personnel) AND
                     (newjobs.curr_ssn <> personnel.ssn)
                     do
                     begin
                        NEXT(personnel);
                     end;
                  WRITELN('    extra person,
                          ',personnel.rate,
                         personnel.last_name,'assign to ',
                         newjobs.pros_ssn);
                  WRITELN;
               end;
            NEXT(newjobs);
         end;

      close(newjobs);
      close(personnel);
      close(jobsfile);
   end;
```

```
program REENLIST(RECEIVE ssn       : string(11);
                 RETURN  intention : boolean);

var
    re_enlist : record
       ssn    : string(11);
       intent : boolean;
    end;

    personnel : record
       last_name   : string(15);
       first_name  : string(15);
       middle_nam  : string(15);
       ssn         : string(11);
       rate        : string(5);
    end;

    r_intent : char;

begin
   open(re_enlist, 'reenlist');
   goto(1, re_enlist);
   while NOT EOF(re_enlist) AND
      ((ssn <> re_enlist.ssn) OR
      (re_enlist.deleted = true))  do
      begin
         NEXT(re_enlist);
      end;
   if (ssn = re_enlist.ssn)
      then
         begin
           intention := re_enlist.intent;
         end
      else
         begin
            open(personnel, 'persinfo');
            goto(1, personnel);
            while NOT EOF(personnel) AND
                  (ssn <> personnel.ssn) do
               begin
                  NEXT(personnel);
               end;
            CLEAR;
            WRITELN;
            WRITELN;
            WRITELN;
            WRITELN('Does  ', personnel.rate,' ',
                     personnel.first_name,
                     personnel.last_name,
                    '  intend to reenlist?  (Y/N)');
```

```
              r_intent := ' ';

              repeat
                 begin
                    READLN(r_intent);
                 end;
              until (r_intent = 'Y') OR (r_intent = 'y')
                    OR (r_intent = 'N')
                    OR (r_intent = 'n');

              if (r_intent = 'Y') OR (r_intent = 'y')
                 then
                    begin
                       intention := true;
                    end

                 else

                    begin
                       intention := false;
                    end;

               re_enlist.intent := intention;
               re_enlist.ssn := ssn;
               append(re_enlist);
               close(re_enlist);
            end;

      close(personnel);
   end;
```

```
program WORK_CTR(RECEIVE billet_nr : integer;
                 RETURN  work_cntr : string(3);
                         job_lngth : integer);

var
    jobsfile : record
       billet_nr   : real;
       job_descrp  : string(20);
       workcenter  : string(3);
       min_paygr   : string(2);
       max_paygr   : string(2);
       rating      : string(3);
       curr_pers   : string(11);
       pros_pers   : string(11);
       job_length  : real;
       nec_reqd    : boolean;
       nec         : real;
    end;

begin
   open(jobsfile, 'jobs');
   goto(1, jobsfile);
   while NOT EOF(jobsfile) AND
         ((jobsfile.deleted = true) OR
         (jobsfile.billet_nr <> FLOAT(billet_nr))) do
      begin
         NEXT(jobsfile);
      end;

   if EOF(jobsfile) then
      begin
         work_cntr := '   ';
         job_lngth := 0;
      end

   else

      begin
         work_cntr := jobsfile.workcenter;
         job_lngth := ROUND(jobsfile.job_length);
      end;

   close(jobsfile);
end;
```

```
program WRITE_IT(RECEIVE billet_nr  : integer;
                         assign_ssn : string(11);
                         future_ssn : string(11));

var
    assign : record
       billet_nr : real;
       curr_ssn  : string(11);
       pros_ssn  : string(11);
    end;

begin
   open(assign, 'new_jobs');
   assign.billet_nr := FLOAT(billet_nr);
   assign.curr_ssn := assign_ssn;
   assign.pros_ssn := future_ssn;
   append(assign);
   close(assign);
end;
```

APPENDIX D

TURBO PASCAL PROGRAMS

```
program get_date ;

type

  registers = record
    ax,bx,cx,dx,bp,si,di,ds,es,fg : integer;
  end;

VAR

  RECPACK       : registers; {record for MsDos call}
  MONTH, DAY    : STRING[2];
  YEAR          : STRING[4];

  err, iMONTH, iDAY, iYEAR, DX, CX : INTEGER;
  date, date3Ø  : string[8];

{$I ASCIIPRM.PAS } {This is INSIGHT 2+ Pascal paramter
                    passing source }

BEGIN
  WITH RECPACK DO BEGIN
    AX := $2A SHL 8;
  END;
  MSDOS (RECPACK);               { call function }
  WITH RECPACK DO BEGIN
    STR (CX, YEAR);
    STR (DX MOD 256, DAY);
    STR (DX SHR 8, MONTH);
  END;

  val (month, imonth, err);
  val (day, iday, err);
  val (year, iyear, err);

  if imonth <= 9
    then
       MONTH := 'Ø'+MONTH;
  if iday <= 9
     then
        DAY := 'Ø'+DAY;

  DATE := YEAR+MONTH+DAY;
```

```
iday := iday + 30;

if ((imonth=1) or (imonth=3) or (imonth=5) or
   (imonth=7) or (imonth=8) or (imonth=10)) and
   (iday>31)  then
   begin
      imonth := imonth + 1;
      iday := iday - 31;
   end;

if (imonth=2) and (iday>28) then
   begin
      imonth := imonth + 1;
      iday := iday - 28;
   end;

if ((imonth=4) or (imonth=6) or (imonth=9) or
   (imonth=11)) and (iday>30)  then
   begin
      imonth := imonth + 1;
      iday := iday - 30;
   end;

if (imonth=12) and (iday>31) then
   begin
      iyear := iyear + 1;
      imonth := 1;
      iday := iday - 31;
   end;

str(iyear:4,year);

if imonth <= 9
   then
      begin
         str(imonth:1,month);
         MONTH := '0'+MONTH;
      end
   else
      begin
         str(imonth:2,month)
      end;

if iday <= 9
   then
      begin
         str(iday:1,day);
         DAY := '0'+DAY;
      end
   else
```

```
            begin
               str(iday:2,day);
            end;

      DATE3Ø := YEAR+MONTH+DAY;
      INIT_PARAM_ADDR;
      RESET_PARAM_MEMORY (2);
      WRITE_STRING(date);
      WRITE_STRING(date3Ø);

   END.
```

```
program job_time;

{$v-,I-}

VAR
    start_date,
    today_date   : string[8];
    job_length   : real;
    exceeded,
    exceed30     : boolean;

    istrtyr,
    istrtmo,
    istrtda,
    itdayyr,
    itdaymo,
    itdayda,
    err,
    jobtime      : integer;

    start_yr,
    today_yr     : string[4];

    start_mo,
    today_mo,
    start_da,
    today_da     : string[2];

{$I ASCIIPRM.PAS} {This is INSIGHT 2+ Pascal paramter
                   passing source }

begin
    INIT_PARAM_ADDR;

    READ_STRING(start_date);
    READ_STRING(today_date);
    READ_REAL(job_length);

    start_yr := COPY(start_date,1,4);
    start_mo := COPY(start_date,5,2);
    start_da := COPY(start_date,7,2);

    today_yr := COPY(today_date,1,4);
    today_mo := COPY(today_date,5,2);
    today_da := COPY(today_date,7,2);

    val(start_yr,istrtyr,err);
    val(start_mo,istrtmo,err);
    val(start_da,istrtda,err);

    val(today_yr,itdayyr,err);
    val(today_mo,itdaymo,err);
    val(today_da,itdayda,err);

    if itdayda < istrtda then
       begin
          itdayda := itdayda + 30;
          itdaymo := itdaymo - 1;
       end;

    if itdaymo < istrtmo then
       begin
          itdaymo := itdaymo + 12;
          itdayyr := itdayyr - 1;
       end;

    jobtime := ((itdayyr - istrtyr) * 365) +
               ((itdaymo - istrtmo) * 30) +
               (itdayda - istrtda);

    if jobtime >= job_length
       then
          exceeded := true
       else
          exceeded := false;

    if (jobtime + 30) >= job_length
       then
          exceed30 := true
       else
          exceed30 := false;

    RESET_PARAM_MEMORY (1);

    WRITE_BOOLEAN(exceeded);
    WRITE_BOOLEAN(exceed30);

END.
```

APPENDIX E

KNOWLEDGE BASE PROGRAM

```
TITLE AMO DISPLAY

                    AMO  KNOWLEDGE  BASE

                    A S S I S T A N T

                  M A I N T E N A N C E

                      O F F I C E R

      A knowledge base designed to assist the
      squadron Assistant Maintenance Officer
      in assigning maintenance personnel to
      appropriate billets.

      Author:  Thomas P. Alston, LT, USN
      Date:    28 February 1987

      Press Function Key 3 STRT to start AMO.

NUMERIC     billet counter
AND         billet number
AND         job length
AND         personnel counter
AND         current job
AND         previous job
AND         total days in previous job
AND         total tad time
AND         nec code
```

```
STRING     todays date
AND        date plus one month
AND        billet description
AND        work center
AND        minimum paygrade
AND        maximum paygrade
AND        paygrade
AND        desired paygrade
AND        billet rating
AND        persons rating
AND        social security number
AND        current ssn
AND        prospective ssn
AND        date reported
AND        prd
AND        eaos
AND        start date
AND        cdi for work center
AND        striker rating

SIMPLEFACT nec required
AND        nec found
AND        billet_eof
AND        personnel_eof
AND        ssn was found
AND        cdi
AND        qar
AND        plane captain
AND        ordnance certified
AND        striker
AND        person assigned
AND        has exceeded job length
AND        will exceed job length soon
AND        person meets all_requirements
AND        reenlistment intention

FORGET ALL
SUPPRESS ALL
CONFIDENCE OFF
UNKNOWN FAIL

! The goal of the knowledge base is AMO_IS_DONE.

1. AMO_IS_DONE
```

```
RULE 1   Check if goal (AMO_IS_DONE) is reached
IF system initialized
AND all_billets are filled
AND FORGET initialize personnel counter
AND initialize personnel counter
AND all_personnel are assigned
AND CALL printout
THEN AMO_IS_DONE

RULE 2   Initialize new_jobs.dbf and reenlist.dbf and
!        get todays date and date 30 days from today
CALL init_new
ACTIVATE get_date.com
RETURN todays date
RETURN date plus one month
IF billet counter := 1
AND personnel counter := 1
THEN system initialized

! Next four rules are used to check if all billets have
! been filled.  The knowledge base starts with the
! billet counter initialized to 1, and cycles through
! the jobs.dbf file until it reaches the end of the
! file.  Originally, the rules used recursion directly
! on "all_billets are filled" to fill approximately 170
! billets.  However, because Insight 2+ detects
! possible infinite recursion, a method of reducing the
! number of recursive calls had to be implemented.
! Therefore, the rules were modified to allow approx-
! imately 20 billets to be assigned for each level of
! recursion.

RULE 3   Recursive call for all_billets are filled
IF fill some billets
AND FORGET fill some billets
AND fill some billets
AND FORGET fill some billets
AND fill some billets
AND FORGET fill some billets
AND fill some billets
AND FORGET fill some billets
AND all_billets are filled
THEN all_billets are filled

RULE 4   Stopping condition for all_billets are filled
IF billet_eof
THEN all_billets are filled
```

```
RULE 5   Conditions for filling one billet
IF have a billet
AND have a person
AND have a prospective person
AND CALL write_it
SEND billet number
SEND current ssn
SEND prospective ssn
AND FORGET have a billet
AND FORGET have a person
AND FORGET have a prospective person
AND billet counter := billet counter + 1
THEN one billet was filled

RULE 6   Fill some (5) billets, one at time
IF one billet was filled
AND FORGET one billet was filled
AND one billet was filled
AND FORGET one billet was filled
AND one billet was filled
AND FORGET one billet was filled
AND one billet was filled
AND FORGET one billet was filled
AND one billet was filled
AND FORGET one billet was filled
THEN fill some billets

RULE 7   Initialize the personnel counter
IF personnel counter := 1
THEN initialize personnel counter

! Next three rules are used to check if all personnel
! have been assigned.  The knowledge base starts with
! the personnel counter initialized to 1, and cycles
! through the personel.dbf file until it reaches the
! end of the file.  The rules use recursion to check
! if all persons have been assigned.  Again, the
! recursion is modified so that the levels of
! recursion are reduced.

RULE 8   Check if a person has been assigned
IF have ssn
AND person already assigned
AND FORGET have ssn
AND FORGET person already assigned
AND personnel counter := personnel counter + 1
THEN one person is assigned
```

```
RULE 9   Recursive call for all_personnel are assigned
IF one person is assigned
AND FORGET one person is assigned
AND one person is assigned
AND FORGET one person is assigned
AND one person is assigned
AND FORGET one person is assigned
AND one person is assigned
AND FORGET one person is assigned
AND all_personnel are assigned
THEN all_personnel are assigned

RULE 10  Stop condition for all_personnel are assigned
IF personnel_eof
THEN all_personnel are assigned

RULE 11  Get a billet
CALL get_bill
SEND billet counter
RETURN billet number
RETURN billet description
RETURN work center
RETURN minimum paygrade
RETURN maximum paygrade
RETURN billet rating
RETURN current ssn
RETURN prospective ssn
RETURN job length
RETURN nec required
RETURN nec code
RETURN billet_eof
IF NOT billet_eof
THEN have a billet

! Rules that conclude "have a person" are listed
! below.  These rules are used to match a billet
! with a person in the personel.dbf file.  The
! knowledge base will try to assign the person who
! is currently filling the billet to that billet
! unless the person is about to leave the squadron
! due to PRD/EAOS considerations, or the person has
! been assigned to another billet, or the person is
! not currently assigned to a billet.
```

```
RULE 12  Check if current person is ok for the billet
IF current ssn > "000-00-0000"
AND social security number := current ssn
AND have the current person
AND prd > date plus one month
AND eaos > date plus one month
AND have job length information
AND NOT has exceeded job length
AND prospective ssn > "000-00-0000"
AND person hasnt been assigned
AND FORGET have the current person
AND FORGET person hasnt been assigned
AND FORGET have job length information
THEN have a person
AND need a prospective person

RULE 13  Check if current person is ok for the billet
IF current ssn > "000-00-0000"
AND social security number := current ssn
AND have the current person
AND prd > date plus one month
AND eaos > date plus one month
AND have job length information
AND NOT has exceeded job length
AND prospective ssn <= "000-00-0000"
AND person hasnt been assigned
AND FORGET have the current person
AND FORGET person hasnt been assigned
AND FORGET have job length information
AND FORGET need a prospective person
THEN have a person

RULE 14  Check if current person is ok for the billet
IF current ssn > "000-00-0000"
AND social security number := current ssn
AND have the current person
AND prd > date plus one month
AND eaos <= date plus one month
AND intends to reenlist
AND have job length information
AND NOT has exceeded job length
AND prospective ssn > "000-00-0000"
AND person hasnt been assigned
AND FORGET have the current person
AND FORGET intends to reenlist
AND FORGET person hasnt been assigned
AND FORGET have job length information
THEN have a person
AND need a prospective person
```

```
RULE 15  Check if current person is ok for the billet
IF current ssn > "000-00-0000"
AND social security number := current ssn
AND have the current person
AND prd > date plus one month
AND eaos <= date plus one month
AND intends to reenlist
AND have job length information
AND NOT has exceeded job length
AND prospective ssn <= "000-00-0000"
AND person hasnt been assigned
AND FORGET need a prospective person
AND FORGET have the current person
AND FORGET intends to reenlist
AND FORGET person hasnt been assigned
AND FORGET have job length information
THEN have a person

RULE 16  Check if current person is ok for the billet
IF current ssn > "000-00-0000"
AND social security number := current ssn
AND have the current person
AND prd > todays date
AND eaos > todays date
AND prd <= date plus one month
AND person hasnt been assigned
AND FORGET have the current person
AND FORGET person hasnt been assigned
THEN have a person
AND need a prospective person

RULE 17  Check if prospective person is ok for billet
IF current ssn > "000-00-0000"
AND prospective ssn > "000-00-0000"
AND social security number := prospective ssn
AND FORGET have the current person
AND have the current person
AND prd > date plus one month
AND eaos > date plus one month
AND person hasnt been assigned
AND FORGET need a prospective person
AND FORGET have the current person
AND FORGET person hasnt been assigned
THEN have a person
```

```
RULE 18  Check if prospective person is ok for billet
IF current ssn > "000-00-0000"
AND prospective ssn > "000-00-0000"
AND social security number := prospective ssn
AND FORGET have the current person
AND have the current person
AND prd > date plus one month
AND eaos <= date plus one month
AND person hasnt been assigned
AND FORGET person hasnt been assigned
AND intends to reenlist
AND FORGET intends to reenlist
AND FORGET have the current person
AND FORGET need a prospective person
THEN have a person

RULE 19  Find a person to fill billet
IF initialize personnel counter
AND initialize desired paygrade
AND desired paygrade < minimum paygrade
AND FORGET initialize personnel counter
AND FORGET initialize desired paygrade
AND FORGET find a person
AND find a person
AND FORGET person hasnt been assigned
AND FORGET need a prospective person
AND FORGET initialize personnel counter
AND FORGET initialize desired paygrade
AND FORGET have a possible person
THEN have a person

RULE 20  Find a person to fill billet
IF initialize personnel counter
AND initialize desired paygrade
AND have a possible person
AND nec required
AND person has nec
AND start date <= todays date
AND prd > date plus one month
AND eaos > date plus one month
AND person hasnt been assigned
AND current ssn := social security number
AND FORGET person hasnt been assigned
AND FORGET initialize personnel counter
AND FORGET initialize desired paygrade
AND FORGET have a possible person
AND FORGET person has nec
AND FORGET need a prospective person
THEN have a person
```

```
RULE 21  Find a person to fill billet
IF initialize personnel counter
AND initialize desired paygrade
AND have a possible person
AND nec required
AND person has nec
AND start date <= todays date
AND prd > date plus one month
AND eaos <= date plus one month
AND person hasnt been assigned
AND FORGET person hasnt been assigned
AND intends to reenlist
AND current ssn := social security number
AND FORGET initialize personnel counter
AND FORGET initialize desired paygrade
AND FORGET have a possible person
AND FORGET need a prospective person
AND FORGET person has nec
AND FORGET intends to reenlist
THEN have a person

RULE 22  Find a person to fill billet
IF initialize personnel counter
AND initialize desired paygrade
AND have a possible person
AND production work center
AND FORGET production work center
AND cdi
AND start date <= todays date
AND prd > date plus one month
AND eaos >  date plus one month
AND person hasnt been assigned
AND current ssn := social security number
AND FORGET person hasnt been assigned
AND FORGET initialize personnel counter
AND FORGET initialize desired paygrade
AND FORGET have a possible person
AND FORGET need a prospective person
THEN have a person
```

```
RULE 23  Find a person to fill billet
IF initialize personnel counter
AND initialize desired paygrade
AND have a possible person
AND production work center
AND FORGET production work center
AND cdi
AND start date <= todays date
AND prd > date plus one month
AND eaos <= date plus one month
AND intends to reenlist
AND person hasnt been assigned
AND current ssn := social security number
AND FORGET person hasnt been assigned
AND FORGET initialize personnel counter
AND FORGET initialize desired paygrade
AND FORGET have a possible person
AND FORGET need a prospective person
AND FORGET intends to reenlist
THEN have a person

RULE 24  Find a person to fill billet
IF initialize personnel counter
AND initialize desired paygrade
AND have a possible person
AND work center = "Ø4Ø"
AND cdi
AND start date <= todays date
AND prd > date plus one month
AND eaos >  date plus one month
AND person hasnt been assigned
AND current ssn := social security number
AND FORGET person hasnt been assigned
AND FORGET initialize personnel counter
AND FORGET initialize desired paygrade
AND FORGET have a possible person
AND FORGET need a prospective person
THEN have a person
```

```
RULE 25  Find a person to fill billet
IF initialize personnel counter
AND initialize desired paygrade
AND have a possible person
AND work center = "Ø4Ø"
AND cdi
AND start date <= todays date
AND prd > date plus one month
AND eaos <= date plus one month
AND intends to reenlist
AND person hasnt been assigned
AND current ssn := social security number
AND FORGET person hasnt been assigned
AND FORGET initialize personnel counter
AND FORGET initialize desired paygrade
AND FORGET have a possible person
AND FORGET need a prospective person
AND FORGET intends to reenlist
THEN have a person

RULE 26  Find a person to fill billet
IF initialize personnel counter
AND initialize desired paygrade
AND have a possible person
AND work center = "23Ø"
AND ordnance certified
AND start date <= todays date
AND prd > date plus one month
AND eaos > date plus one month
AND person hasnt been assigned
AND current ssn := social security number
AND FORGET person hasnt been assigned
AND FORGET initialize personnel counter
AND FORGET initialize desired paygrade
AND FORGET have a possible person
AND FORGET need a prospective person
THEN have a person
```

```
RULE 27  Find a person to fill billet
IF initialize personnel counter
AND initialize desired paygrade
AND have a possible person
AND work center = "230"
AND ordnance certified
AND start date <= todays date
AND prd > date plus one month
AND eaos <= date plus one month
AND person hasnt been assigned
AND FORGET person hasnt been assigned
AND intends to reenlist
AND current ssn := social security number
AND FORGET initialize personnel counter
AND FORGET initialize desired paygrade
AND FORGET have a possible person
AND FORGET need a prospective person
AND FORGET intends to reenlist
THEN have a person

RULE 28  Find a person to fill billet
IF initialize personnel counter
AND initialize desired paygrade
AND have a possible person
AND work center = "310"
AND plane captain
AND start date <= todays date
AND prd > date plus one month
AND eaos > date plus one month
AND person hasnt been assigned
AND current ssn := social security number
AND FORGET person hasnt been assigned
AND FORGET initialize personnel counter
AND FORGET initialize desired paygrade
AND FORGET have a possible person
AND FORGET need a prospective person
THEN have a person

RULE 29  Continue looking for a person
IF FORGET have a possible person
AND personnel counter := personnel counter + 1
AND current ssn := "000-00-0000"
AND have a person
THEN have a person
```

```
RULE 30  Is work center a production work center
IF work center = "110"
OR work center = "120"
OR work center = "12C"
OR work center = "13A"
OR work center = "13B"
OR work center = "210"
OR work center = "220"
OR work center = "230"
OR work center = "260"
OR work center = "280"
THEN production work center

! Next rules conclude "find a person" is used when the
! search could not find a person who meets all
! requirements, so another search is conducted to find
! a person who meets the minimum requirements.

RULE 31  Find a person to fill billet
IF initialize personnel counter
AND initialize desired paygrade
AND have a possible person
AND desired paygrade < minimum paygrade
AND current ssn := "none found"
AND social security number := "none found"
AND FORGET initialize personnel counter
AND FORGET initialize desired paygrade
AND FORGET have a possible person
THEN find a person
AND need a prospective person

RULE 32  Find a person to fill billet
IF initialize personnel counter
AND initialize desired paygrade
AND have a possible person
AND start date <= todays date
AND prd > date plus one month
AND eaos > date plus one month
AND person hasnt been assigned
AND FORGET person hasnt been assigned
AND prospective ssn <= "000-00-0000"
AND current ssn := social security number
AND FORGET initialize personnel counter
AND FORGET initialize desired paygrade
AND FORGET have a possible person
AND FORGET need a prospective person
THEN find a person
```

```
RULE 33  Find a person to fill billet
IF initialize personnel counter
AND initialize desired paygrade
AND have a possible person
AND NOT nec required
AND start date <= todays date
AND prd > date plus one month
AND eaos > date plus one month
AND person hasnt been assigned
AND current ssn := social security number
AND FORGET person hasnt been assigned
AND FORGET initialize personnel counter
AND FORGET initialize desired paygrade
AND FORGET have a possible person
AND FORGET need a prospective person
THEN find a person

RULE 34  Continue looking for a person
IF FORGET have a possible person
AND personnel counter := personnel counter + 1
AND current ssn := "000-00-0000"
AND find a person
THEN find a person

! Rule that concludes "person has nec"

RULE 35  Check if person has the required nec
CALL checknec
SEND social security number
SEND nec code
RETURN nec found
IF nec found
THEN person has nec

! Rules that conclude "have a prospective person"

RULE 36  Find a prospective person
IF NOT need a prospective person
AND prospective ssn := "           "
AND FORGET need a prospective person
THEN have a prospective person
```

```
RULE 37  Find a prospective person
IF prospective ssn > "000-00-0000"
AND social security number := prospective ssn
AND FORGET have the current person
AND have the current person
AND prd > date plus one month
AND eaos > date plus one month
AND person hasnt been assigned
AND FORGET person hasnt been assigned
AND current ssn <> social security number
AND FORGET have the current person
THEN have a prospective person

RULE 38  Find a prospective person
IF initialize personnel counter
AND initialize desired paygrade
AND have a possible person
AND desired paygrade < minimum paygrade
AND prospective ssn := "none found"
AND social security number := "none found"
AND FORGET initialize personnel counter
AND FORGET initialize desired paygrade
AND FORGET have a possible person
THEN have a prospective person

RULE 39  Find a prospective person
IF initialize personnel counter
AND initialize desired paygrade
AND have a possible person
AND nec required
AND person has nec
AND prd > date plus one month
AND eaos > date plus one month
AND current ssn <> social security number
AND person hasnt been assigned
AND FORGET person hasnt been assigned
AND FORGET person has nec
THEN have a prospective person

RULE 40  Find a prospective person
IF initialize personnel counter
AND initialize desired paygrade
AND have a possible person
AND work center = "040"
AND billet rating = "ANY"
AND prd > date plus one month
AND eaos > date plus one month
AND current ssn <> social security number
AND person hasnt been assigned
AND FORGET person hasnt been assigned
THEN have a prospective person
```

```
RULE 41  Find a prospective person
IF initialize personnel counter
AND initialize desired paygrade
AND have a possible person
AND work center = "040"
AND qar
AND prd > date plus one month
AND eaos > date plus one month
AND current ssn <> social security number
AND person hasnt been assigned
AND FORGET person hasnt been assigned
THEN have a prospective person

RULE 42  Find a prospective person
IF initialize personnel counter
AND initialize desired paygrade
AND have a possible person
AND work center = "040"
AND billet rating = "AD"
AND have a possible person
AND cdi
AND cdi for work center = "110"
AND prd > date plus one month
AND eaos > date plus one month
AND current ssn <> social security number
AND person hasnt been assigned
AND FORGET person hasnt been assigned
THEN have a prospective person

RULE 43  Find a prospective person
IF initialize personnel counter
AND initialize desired paygrade
AND have a possible person
AND work center = "040"
AND billet rating = "AE"
AND cdi
AND cdi for work center = "220"
AND prd > date plus one month
AND eaos > date plus one month
AND current ssn <> social security number
AND person hasnt been assigned
AND FORGET person hasnt been assigned
THEN have a prospective person
```

```
RULE 44  Find a prospective person
IF initialize personnel counter
AND initialize desired paygrade
AND have a possible person
AND work center = "040"
AND billet rating = "AME"
AND cdi
AND cdi for work center = "13B"
AND prd > date plus one month
AND eaos > date plus one month
AND current ssn <> social security number
AND person hasnt been assigned
AND FORGET person hasnt been assigned
THEN have a prospective person

RULE 45  Find a prospective person
IF initialize personnel counter
AND initialize desired paygrade
AND have a possible person
AND work center = "040"
AND billet rating = "AMH"
AND cdi
AND cdi for work center = "120"
AND prd > date plus one month
AND eaos > date plus one month
AND current ssn <> social security number
AND person hasnt been assigned
AND FORGET person hasnt been assigned
THEN have a prospective person

RULE 46  Find a prospective person
IF initialize personnel counter
AND initialize desired paygrade
AND have a possible person
AND work center = "040"
AND billet rating = "AQ"
AND cdi
AND cdi for work center = "280"
AND prd > date plus one month
AND eaos > date plus one month
AND current ssn <> social security number
AND person hasnt been assigned
AND FORGET person hasnt been assigned
THEN have a prospective person
```

```
RULE 47  Find a prospective person
IF initialize personnel counter
AND initialize desired paygrade
AND have a possible person
AND work center = "040"
AND billet rating = "AO"
AND cdi
AND cdi for work center = "230"
AND prd > date plus one month
AND eaos > date plus one month
AND current ssn <> social security number
AND person hasnt been assigned
AND FORGET person hasnt been assigned
THEN have a prospective person

RULE 48  Find a prospective person
IF initialize personnel counter
AND initialize desired paygrade
AND work center = "310"
AND have a possible person
AND plane captain
AND prd > date plus one month
AND eaos > date plus one month
AND current ssn <> social security number
AND person hasnt been assigned
AND FORGET person hasnt been assigned
AND have job length information
AND NOT has exceeded job length
THEN have a prospective person

RULE 49  Find a prospective person
IF initialize personnel counter
AND initialize desired paygrade
AND work center = "230"
AND have a possible person
AND ordnance certified
AND prd > date plus one month
AND eaos > date plus one month
AND current ssn <> social security number
AND person hasnt been assigned
AND FORGET person hasnt been assigned
THEN have a prospective person
```

```
RULE 50  Find a prospective person
IF initialize personnel counter
AND initialize desired paygrade
AND have a possible person
AND cdi
AND cdi for work center = work center
AND prd > date plus one month
AND eaos > date plus one month
AND current ssn <> social security number
AND person hasnt been assigned
AND FORGET person hasnt been assigned
THEN have a prospective person

RULE 51  Find a prospective person
IF FORGET have a possible person
AND personnel counter := personnel counter + 1
AND prospective ssn := "000-00-0000"
AND have a prospective person
THEN have a prospective person

! Next rules conclude "have a possible person"

RULE 52  Check if no one has been found
IF desired paygrade < minimum paygrade
THEN have a possible person
```

```
RULE 53  Call GET_PERS.PAS
CALL get_pers
SEND personnel counter
SEND billet rating
SEND desired paygrade
RETURN ssn was found
RETURN social security number
RETURN persons rating
RETURN paygrade
RETURN date reported
RETURN prd
RETURN eaos
RETURN current job
RETURN start date
RETURN previous job
RETURN total days in previous job
RETURN cdi
RETURN cdi for work center
RETURN qar
RETURN plane captain
RETURN ordnance certified
RETURN striker
RETURN striker rating
RETURN total tad time
RETURN personnel counter
IF ssn was found
THEN have a possible person

RULE 54  If person wasn't found, then use recursion to
! find a person at the next lower paygrade
IF NOT ssn was found
AND CALL next_pg
SEND desired paygrade
RETURN desired paygrade
AND personnel counter := 1
AND have a possible person
THEN have a possible person

RULE 55  Initialize "desired paygrade"
IF desired paygrade := maximum paygrade
AND FORGET person meets all_requirements
THEN initialize desired paygrade

RULE 56  Assert person meets all_requirements
IF 1 = 1
THEN assert person meets all_requirements
AND person meets all_requirements

RULE 57
IF Ø = 1
THEN person meets all_requirements
```

```
RULE 58
IF Ø = 1
THEN need a prospective person

RULE 59  Check if person has been in job too long
ACTIVATE job_time.com
SEND start date
SEND todays date
SEND job length
RETURN has exceeded job length
RETURN will exceed job length soon
THEN have job length information

RULE 6Ø  Get a person when you have a ssn
CALL get_pssn
SEND social security number
RETURN ssn was found
RETURN persons rating
RETURN paygrade
RETURN date reported
RETURN prd
RETURN eaos
RETURN current job
RETURN start date
RETURN previous job
RETURN total days in previous job
RETURN cdi
RETURN cdi for work center
RETURN qar
RETURN plane captain
RETURN ordnance certified
RETURN striker
RETURN striker rating
RETURN total tad time
RETURN personnel counter
IF ssn was found
THEN have the current person

RULE 61  Call CHECK_NJ.PAS
CALL check_nj
SEND social security number
RETURN person assigned
IF NOT person assigned
THEN person hasnt been assigned
```

```
RULE 62  Call REENLIST.PAS
CALL reenlist
SEND social security number
RETURN reenlistment intention
IF reenlistment intention
THEN intends to reenlist

! The next three rules are used to satisfy
! "all_personnel assigned".  The rules get a
! social security number from the personel.dbf
! file and then check to see if it is in the
! new_jobs.dbf file.  If not, then it assigns
! that person to a work center.

RULE 63  Get ssn to check if person has been assigned
CALL get_ssn
SEND personnel counter
RETURN social security number
RETURN personnel_eof
IF NOT personnel_eof
THEN have ssn

RULE 64  Check if person is already assigned
CALL check_nj
SEND social security number
RETURN person assigned
IF person assigned
THEN person already assigned
ELSE assign person

RULE 65  Check if person has already checked out
! of the squadron
IF assign person
AND get the persons data
AND prd <= todays date
AND person has checked out
AND FORGET assign person
AND FORGET get the persons data
AND FORGET person has checked out
THEN person already assigned
```

```
RULE 66  Assign a person when all billets are filled
IF assign person
AND get the persons data
AND work center found
AND CALL write_it
SEND billet number
SEND social security number
SEND work center
AND FORGET assign person
AND FORGET get the persons data
AND FORGET work center found
THEN person already assigned

! This rule is invoked after all billets have been
! assigned and there are "extra" persons in the
! personel.dbf file who need to be assigned to a
! work center.

RULE 67  Get the "extra" person's data
CALL get_prsn
SEND personnel counter
RETURN ssn was found
RETURN social security number
RETURN persons rating
RETURN paygrade
RETURN date reported
RETURN prd
RETURN eaos
RETURN current job
RETURN start date
RETURN previous job
RETURN total days in previous job
RETURN cdi
RETURN cdi for work center
RETURN qar
RETURN plane captain
RETURN ordnance certified
RETURN striker
RETURN striker rating
RETURN total tad time
IF ssn was found
THEN get the persons data

! The following rules' conclusion is "work center
! found", for assigning "extra" personnel.  The
! person will be assigned to his current work center
! if he meets the criteria in the next 3 rules; and
! the other rules determine an appropriate work center
! based on rating and paygrade.
```

```
RULE 68  Assign him to his current work center
IF current job > Ø
AND prd > date plus one month
AND eaos > date plus one month
AND get a work center
AND not_a tad billet
AND have job length information
AND NOT has exceeded job length
AND billet number := 9999
AND FORGET have job length information
AND FORGET get a work center
AND FORGET not_a tad billet
THEN work center found

RULE 69  Assign him to his current work center
IF current job <> Ø
AND prd > date plus one month
AND eaos <= date plus one month
AND intends to reenlist
AND get a work center
AND not_a tad billet
AND have job length information
AND NOT has exceeded job length
AND billet number := 9999
AND FORGET intends to reenlist
AND FORGET have job length information
AND FORGET get a work center
AND FORGET not_a tad billet
THEN work center found

RULE 7Ø  Assign him to his current work center
IF current job > Ø
AND prd <= date plus one month
AND get a work center
AND not_a tad billet
AND have job length information
AND NOT has exceeded job length
AND billet number := 9999
AND FORGET have job length information
AND FORGET get a work center
AND FORGET not_a tad billet
THEN work center found

RULE 71  Person hasn't arrived on board yet
IF start date > date plus one month
AND billet number := 9999
AND work center := start date
THEN work center found
```

```
RULE 72  Assign to Material Control
IF persons rating = "AK"
AND billet number := 9999
AND work center := "Ø5Ø"
THEN work center found

RULE 73  Assign to Material Control
IF paygrade <= "E3"
AND striker
AND striker rating = "AK"
AND billet number := 9999
AND work center := "Ø5Ø"
THEN work center found

RULE 74  Assign to Maintenance Control
IF persons rating = "AZ"
AND billet number := 9999
AND work center := "Ø2Ø"
THEN work center found

RULE 75  Assign to Maintenance Control
IF paygrade <= "E3"
AND striker
AND striker rating = "AZ"
AND billet number := 9999
AND work center := "Ø2Ø"
THEN work center found

RULE 76  Assign to Corrosion Control
IF paygrade <= "E3"
AND persons rating = "AMS"
AND billet number := 9999
AND work center := "12C"
THEN work center found

RULE 77  Assign to Corrosion Control
IF paygrade <= "E3"
AND striker
AND striker rating = "AMS"
AND billet number := 9999
AND work center := "12C"
THEN work center found

RULE 78  Assign to Line Division
IF paygrade <= "E3"
AND billet number := 9999
AND work center := "31Ø"
THEN work center found
```

```
RULE 79  Assign to Power Plants
IF paygrade >= "E7"
AND persons rating = "AD"
AND billet number := 9999
AND work center := "100"
THEN work center found

RULE 80  Assign to Power Plants
IF paygrade >= "E4"
AND persons rating = "AD"
AND billet number := 9999
AND work center := "110"
THEN work center found

RULE 81  Assign to Airframes
IF persons rating = "AM"
AND billet number := 9999
AND work center := "100"
THEN work center found

RULE 82  Assign to Airframes
IF paygrade >= "E4"
AND persons rating = "AMS"
AND billet number := 9999
AND work center := "120"
THEN work center found

RULE 83  Assign to Airframes
IF paygrade >= "E4"
AND persons rating = "AMH"
AND billet number := 9999
AND work center := "120"
THEN work center found

RULE 84  Assign to PR Branch
IF paygrade >= "E4"
AND persons rating = "PR"
AND billet number := 9999
AND work center := "13A"
THEN work center found

RULE 85  Assign to AME Branch
IF paygrade >= "E4"
AND persons rating = "AME"
AND billet number := 9999
AND work center := "13B"
THEN work center found
```

```
RULE 86  Assign to Avionics
IF paygrade >= "E7"
AND persons rating = "AT"
AND billet number := 9999
AND work center := "200"
THEN work center found

RULE 87  Assign to Avionics
IF paygrade >= "E7"
AND persons rating = "AE"
AND billet number := 9999
AND work center := "200"
THEN work center found

RULE 88  Assign to Avionics
IF paygrade >= "E7"
AND persons rating = "AQ"
AND billet number := 9999
AND work center := "200"
THEN work center found

RULE 89  Assign to Avionics
IF paygrade >= "E4"
AND persons rating = "AT"
AND billet number := 9999
AND work center := "210"
THEN work center found

RULE 90  Assign to Avionics
IF paygrade >= "E4"
AND persons rating = "AE"
AND billet number := 9999
AND work center := "220"
THEN work center found

RULE 91  Assign to Ordnance
IF paygrade >= "E4"
AND persons rating = "AO"
AND billet number := 9999
AND work center := "230"
THEN work center found

RULE 92  Assign to Radar/Fire Control Branch
IF paygrade >= "E4"
AND persons rating = "AQ"
AND billet number := 9999
AND work center := "260"
THEN work center found
```

```
! This rule gets a work center and job length when the
! current job is provided
RULE 93  Get the work center he is assigned to
CALL work_ctr
SEND current job
RETURN work center
RETURN job length
THEN get a work center

RULE 94  Determine if billet is not a TAD billet
IF work center = "020"
OR work center = "030"
OR work center = "040"
OR work center = "04A"
OR work center = "04C"
OR work center = "050"
OR work center = "100"
OR work center = "110"
OR work center = "120"
OR work center = "12C"
OR work center = "13A"
OR work center = "13B"
OR work center = "140"
OR work center = "200"
OR work center = "210"
OR work center = "220"
OR work center = "230"
OR work center = "260"
OR work center = "280"
OR work center = "300"
OR work center = "310"
OR work center = "320"
THEN not_a tad billet

END
```

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22304-6145	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, California 93943-5002	2
3.	Professor Norman H. Lyons, Code 54Lb Department of Administrative Sciences Naval Postgraduate School Monterey, California 93943-5004	1
4.	Professor Taracad Sivasankaran, Code 54Sj Department of Administrative Sciences Naval Postgraduate School Monterey, California 93943-5004	1
5.	Professor Tung Bui, Code 54Bd Department of Administrative Sciences Naval Postgraduate School Monterey, California 93943-5004	1
7.	LT Thomas P. Alston VFA-132 Cecil Field, Florida 32215	2